ELEX MEDIA KOMPUTINDO

SERI APLIKASI PEMROGRAMAN

Pemrograman Animasi dan Game Profesional 2

Agustinus Nalwan

Disertai DISKET

SERI APLIKASI PEMROGRAMAN

Pemrograman Animasi dan Game Profesional

Agustinus Nalwan

Seri Aplikasi Pemrograman

Pemrograman Animasi dan Game Profesional 2

Sanksi Pelanggaran Pasal 44:
Undang-undang Nomor 7 Tahun 1987 Tentang
Perubahan atas undang-undang Nomor 6 Tahun 1982
Tentang Hak Cipta

1. Barangsiapa dengan sengaja dan tanpa hak mengumumkan atau memperbanyak suatu ciptaan atau memberi izin untuk itu, dipidana dengan pidana penjara paling lama 7 (tujuh) tahun dan/atau denda paling banyak Rp 100.000.000,- (seratus juta rupiah).
2. Barangsiapa dengan sengaja menyiarkan, memamerkan, mengedarkan, atau menjual kepada umum suatu ciptaan atau barang hasil pelanggaran Hak Cipta sebagaimana dimaksud dalam ayat (1), dipidana dengan pidana penjara paling lama 5 (lima) tahun dan/atau denda paling banyak Rp 50.000.000,- (lima puluh juta rupiah).

Seri Aplikasi Pemrograman

Pemrograman Animasi dan Game Profesional 2

Agustinus Nalwan

Penerbit PT Elex Media Komputindo
Kelompok Gramedia-Jakarta

Seri Aplikasi Pemrograman
Pemrograman Animasi dan Game Profesional 2
Oleh: Agustinus Nalwan
©1996, PT Elex Media Komputindo
Hak cipta dilindungi oleh Undang-undang
Diterbitkan pertama kali oleh
Penerbit PT Elex Media Komputindo
Kelompok Gramedia, Anggota IKAPI, Jakarta 1996

12196157
ISBN 979-637-830-2

Dilarang mengutip, memperbanyak, atau menerjemahkan sebagian atau seluruh isi buku ini tanpa izin tertulis dari Penerbit.

Dicetak oleh Percetakan Gramedia Jakarta
Isi di luar tanggung jawab Percetakan

DAFTAR ISI

PENDAHULUAN vii

BAB 1 PEMBENAHAN PERTAMA VGACAD.TPU 1
Peletakan Sprite dengan Batasan Layar 2
Peletakan Sprite Satu warna 17
Teknik Peletakan Sprite Transparant 22
Teknik Peletakan Sprite Resize 40

BAB 2 PEMBENAHAN KEDUA VGACAD.TPU 63
Teknik Color Compression 64
Teknik Zero Compression 72
Teknik RLL Compression 80

BAB 3 PEMBENAHAN KETIGA VGACAD.TPU 89
Penggeseran layar dengan Horizontal Scrolling 90
Penggeseran layar dengan Parallax Scrolling 109

BAB 4 MODIFIKASI INTERRUPT KEYBOARD 133

BAB 5 TEKNIK MOMENTUM DAN TABRAKAN 145
Teknik Gerakan Momentum 146
Teknik Tabrakan dengan Pendeteksian Pixel 157
Teknik Tabrakan dengan Pendeteksian Vektor 169

BAB 6 ULASAN TENTANG VGACAD.TPU 191

BAB 7 PEMBUATAN GAME BUG HUNTER 243
Perencanaan 244
Pembuatan Mission Editor 248
Pembuatan Game Engine 279

INFORMASI DAFTAR FILE 359
PENUTUP 361

PENDAHULUAN

Sebelum Anda membaca buku ini, sebaiknya Anda telah menguasai teknik-teknik yang ada pada buku seri pertama *SERI APLIKASI PEMROGRAMAN* **PEMROGRAMAN GAME DAN ANIMASI PROFESIONAL**, karena buku ini merupakan kelanjutan dari seri tersebut. Adapun compiler yang akan kita gunakan tetap seperti yang dulu yaitu kerja sama antara Turbo Pascal 7.00 dan Assembly.

Selamat berjumpa kembali dalam pembuatan game dan animasi profesional. Setelah seri pertama di mana kita mempelajari teknik pengolahan graphic pada mode 320 x 200 256 warna, kini kita beranjak ke tingkat yang lebih sulit. Yang namanya lebih sulit, tentu saja lebih canggih, jadi janganlah cepat berputus asa. Apabila Anda telah menguasai teknik yang diajarkan pada buku seri yang pertama, saya jamin Anda tidak akan mengalami kesulitan dalam mempelajari teknik-teknik di buku ini.

Baiklah, akan saya berikan sedikit gambaran tentang apa yang akan kita pelajari pada seri kedua ini. Pertama-tama kita akan melakukan pembenahan-pembenahan pada unit graphic kita yaitu VGACAD.TPU. Unit ini kalau kita telusuri lebih lanjut masih banyak kelemahan-kelemahannya lho. Tapi semua itu belum dibahas pada buku seri pertama, mengingat buku tersebut masih merupakan dasar pemrograman game. Di sini kita akan memperbaikinya dan menambahkan banyak kemampuan baru yang akan menunjang pembuatan game-game yang lebih handal lagi.

Dengan VGACAD.TPU yang baru ini, Anda bahkan mampu membuat game seperti Super Mario Bross, Prince of Persia, Street Fighter dan lain-lain, bahkan Anda dapat menggunakannya untuk membuat game 3 Dimensi sederhana. Untuk game-game 3 Dimensi yang benar-benar handal akan dibahas pada kesempatan yang akan datang. Game-game tersebut seperti DOOM, Wolfenstein 3D, Wing Comander, Mech Warrior 2 dan lain-lainnya. Dan seperti biasa, pada akhir pembahasan, kita akan mencoba merakit sebuah game dengan memanfaatkan unit graphic VGACAD.TPU yang baru. Proyek yang akan kita buat kali ini ialah game BUG HUNTER yang bertipe SONIC, SUPER MARIO BROSS dan JACK JAZZ RABBIT dengan teknologi yang bisa dibilang melebihi ketiga game ini.

Baiklah, saya tidak akan bicara banyak di sini, yuk kita mulai saja dengan BAB 1.

BAB 1

PEMBENAHAN PERTAMA VGACAD.TPU

PELETAKAN SPRITE DENGAN BATASAN LAYAR

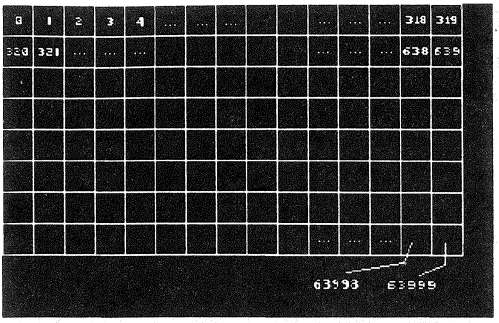
Pembenahan pertama yang akan kita lakukan terhadap VGACAD.TPU adalah pada fungsi PUTIMAGE yang tujuannya untuk meletakkan gambar pada virtual screen atau layar monitor. Apa yang salah pada fungsi ini?

Kalau kita teliti dengan benar, sebenarnya fungsi PUTIMAGE ini sangat vital karena dengan PUTIMAGE ini sebenarnya kita memindahkan data pada sprite yang berisi warna tiap titik yang membentuk sprite tersebut ke alamat memori di mana tempat virtual screen berada. Nah apa yang terjadi kalau kita meletakkan gambar yang melebihi batasan layar 320 x 200 pixel?

Kita ambil contoh sebagai berikut, kita secara tidak sengaja meletakkan spite yang besarnya 50 x 50 titik ke virtual screen (besar virtual screen selalu 320 x 200 titik) pada koordinat (100,190). Tentu saja gambar yang Anda letakkan itu tidak dapat tertampung seluruhnya pada virtual screen. Gambar yang akan kelihatan hanya setinggi sepuluh titik saja kan? Lalu ke mana sisa gambar tersebut akan diletakkan?
Komputer tidak akan memotong gambar tersebut melainkan meletakkannya begitu saja, sehingga sisa gambar tadi akan diletakkan pada alamat di luar memori yang disediakan untuk virtual screen tersebut. Apabila alamat memori tersebut kosong, tidak ada masalah. Bayangkan kalau komputer meletakkan sisa gambar tersebut pada alamat memori di mana kita menyimpan data-data permainan seperti jumlah enerji pesawat tempur, arah gerakan musuh, dan lain lainnya. Game yang kita rancang akan kacau bahkan mungkin saja bisa crashed (macet).

Situasi di atas menggambarkan sprite yang menerjang batas bawah layar. Nah sekarang bagaimana kalau batas kanan atau kiri layar yang diterjang. Bila ada sprite yang menerjang batas kanan atau kiri layar, memori di luar virtual screen tidak akan terganggu, malahan sisa gambar yang melebihi batas layar tersebut akan di tampilkan kembali pada sisi yang berlawanan. Misalnya sprite tersebut menerjang batas kanan layar sehingga hanya kelihatan separuh, maka separuhnya akan kelihatan di sebelah kiri layar. Kenapa hal ini bisa terjadi?

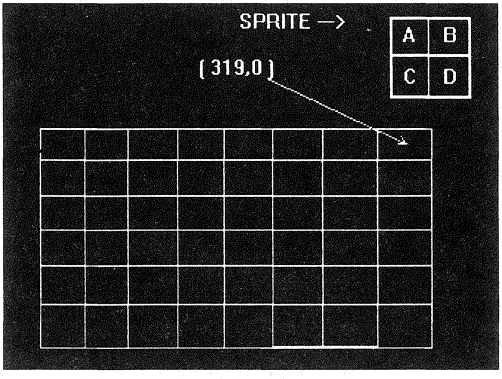
Sebagaimana kita ketahui sistem alokasi pixel pada mode $13 adalah seperti diagram berikut.



Perlu diketahui bahwa jumlah kotak horizontal adalah 320 dan jumlah kotak vertikal adalah 200.

SPRITE YANG MENERJANG BATAS SAMPING LAYAR

Jika kita meletakkan sprite dengan ukuran 2 x 2 pada koordinat (319,0) seperti yang dapat Anda lihat pada gambar di bawah ini, tentu saja sprite tersebut tidak akan cukup diletakkan pada layar.



Kalau kita lihat secara langsung dari diagram ini, titik B dan D tidak akan dapat dialokasikan dengan benar, karena titik B dan D akan berada pada koordinat X yang melebihi 319 (koordinat X maximum).
Tetapi, fungsi PUTIMAGE pada VGACAD.TPU akan meletakkan sprite tersebut dengan proses sebagai berikut:

- Titik A akan diletakkan pada koordinat (319,0) dengan fungsi

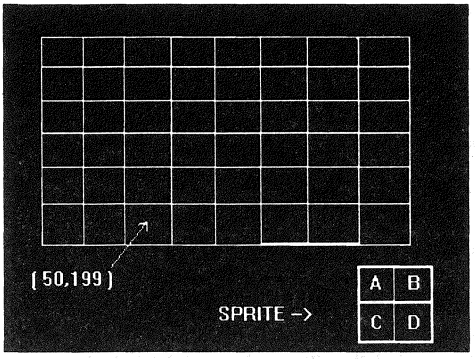
perhitungan offset berarti sama dengan alamat memori layar 0*320+319 = 319. Dalam hal ini titik A berada pada posisi yang benar (lihat pada diagram alokasi titik pada mode $13).

- Titik B akan diletakkan pada koordinat (320,0). Koordinat X maximum adalah 319, tapi karena komputer tidak peduli maka proses perhitungan alamat memori layar tetap dilakukan sebagai berikut 0*320 + 320 = 320. Lihatlah pada diagram alokasi alamat memori pada halaman sebelumnya. Di sana dapat kita lihat bahwa alamat memori 320 itu berada pada lajur paling kiri dari layar. Nah, inilah sebabnya sisa gambar pada kanan layar akan dilanjutkan pada kiri layar.
- Titik C akan diletakkan pada koordinat (319,1) dengan fungsi perhitungan offset berarti sama dengan alamat memori layar 1*320+319 = 639. Titik ini pun berada pada tempat yang benar apabila Anda lihat pada diagram alokasi titik pada mode $13.
- Titik D akan mengalami kejadian yang sama dengan titik B, dia akan ditempatkan pada alamat memori 1*320+320 = 640 yang juga akan berada pada lajur kiri layar.

Dengan ini kita tahu bahwa gambar yang menerjang batas kanan dan kiri layar tidak akan mengganggu alamat memori di luar virtual screen, tapi akan menimbulkan tampilan yang jelek pada layar karena munculnya potongan-potongan gambar pada tempat yang tidak kita inginkan.

SPRITE YANG MENERJANG BATAS BAWAH/ATAS LAYAR

Sekarang lihatlah bagan di bawah ini. Bagan tersebut menggambarkan situasi di mana kita meletakkan sprite berukuran 2 x 2 yang menerjang batas bawah layar.



Tentu saja sprite ini tidak cukup untuk diletakkan pada koordinat (50 , 199) di mana koordinat Y maximum adalah 199. Kalau kita lihat dari diagram, titik C dan D tidak akan dapat dialokasikan dengan benar karena kedua titik tersebut berada pada koordinat Y = 200 (yang tidak mungkin).
Tetapi, PUTIMAGE dari VGACAD.TPU akan melakukan proses sebagai berikut:

- Titik A yang berada pada koordinat (50,199) akan dialokasikan ke 199*320+50 = 63730. Titik ini masih berada di dalam virtual screen (alamat akhir virtual screen = 64000). Bila Anda melihat pada diagram alokasi titik pada mode $13, titik A ini teralokasikan dengan benar.
- Titik B yang berada pada koordinat (51,199) akan dialokasikan ke 199*320+51 = 63631. Titik ini pun masih berada di dalam virtual screen. Dan jika Anda melihat pada diagram alokasi titik pada mode $13, titik B ini teralokasikan dengan benar.
- Titik C yang berada pada koordinat (50,200) akan dialokasikan ke 200*320+50 =64050. Titik ini jelas-jelas telah melebihi batas virtual screen. Titik ini akan dialokasikan ke alamat memori di luar virtual screen yang mungkin saja sedang dipakai untuk penyimpanan data yang lain. Hal ini dapat menyebabkan program Anda crashed (macet).
- Titik D yang berada pada koordinat (50,201) akan dialokasikan ke offset berikutnya yaitu 64051. Titik ini pun berada pada tempat yang tidak semestinya.

Dari sini kita ketahui bahwa sprite yang menerobos batas bawah layar akan mengganggu jalannya program. Bagaimana dengan batas atas layar?

Sprite yang menerobos batas atas layar akan mengalami hal yang sama dengan sprite yang menerobos batas bawah layar. Bedanya, alamat memori yang dialokasi adalah alamat yang lebih di luar virtual screen bagian bawah (mendekati DOS area yang berada pada segment $0000).

Hal-hal sepele seperti inilah yang biasanya menjadi masalah dalam pembuatan game.

PEMECAHAN MASALAH BATASAN LAYAR

Nah sekarang bagaimana kita mengatasi masalah tersebut?

Kita harus mengganti proses PUTIMAGE pada VGACAD.TPU sehingga tidak akan menerobos batas samping maupun atas/bawah layar. Memang teknik croping/pembatasan sprite ini agak rumit bila diterapkan dalam bahasa assembler, tetapi mau tidak mau kita harus melakukannya karena PUTIMAGE inilah yang memberikan pengaruh terbesar pada kecepatan animasi.

Lihatlah procedure PUTIMAGE kita yang baru di bawah ini.

```
Procedure PutImage(Var PicBuff;X1,Y1:Integer;
Var Buffer);Assembler;
Var XWStart:Word;
    XRStart:Word;     { Titik X Pertama yang dibaca }
    YWStart:Word;     { Koordinat Y Pertama untuk penulisan }
    YRStart:Word;     { Titik Y Pertama yang dibaca }
    Length:Word;      { Panjang sprite }
    Width:Word;       { Lebar sprite }
    XRead:Word;       { Jumlah peletakan titik secara }
                      { horizontal }
    YRead:Word;       { Jumlah peletakan titik secara }
                      { vertikal }
  Asm
    Push ds           { Simpan ds ke stack }
    Lds si,Buffer     { Alamatkan DS:SI ke Buffer }
    Les di,PicBuff    { Alamatkan ES:DI ke PicBuff }
    Mov Ax,[ds:si]    { Ambil panjang sprite }
    Mov Length,Ax     { Simpan di Length }
    Mov XRead,Ax      { Jumlah peletakan titik }
                      { horizontal = Length }
    Add si,2          { Arahkan SI ke lebar sprite }
    Mov Ax,[ds:si]    { Ambil lebar sprite }
    Mov Width,Ax      { Simpan di Width }
    Mov YRead,Ax      { Jumlah peletakan titik }
                      { vertikal = Width }
    Mov YRStart,0     { Asumsikan titik vertikal pertama yang }
                      { dibaca ialah titik pertama }
    Mov Ax,Y1         { Ambil koordinat Y di mana sprite akan }
                      { diletakkan }
    Mov YWStart,Ax    { Asumsikan koordinat sprite pada Y }
    Add si,2          { Arahkan SI ke titik pertama sprite }
```

```
Mov XRStart,0     { Asumsikan titik horizontal pertama yang }
                  { dibaca ialah titik pertama }
Mov Ax,X1         { Ambil koordinat X di mana sprite akan }
                  { diletakkan }
Mov XWStart,Ax    { Asumsikan koordinat sprite pada X }
Cmp Ax,0          { Deteksi batas kiri layar }
Jge @X1_Not_Negative
  Neg Ax          { Kalkulasi Pembatasan kiri layar }
  Mov XRStart,Ax
  Mov XWStart,0
  Sub XRead,Ax
@X1_Not_Negative:
  Mov Ax,X1
  Add Ax,Length
  Dec Ax
  Cmp Ax,319
Jle @X1_No_Right_Crop
  Sub Ax,319      { Kalkulasi Pembatasan kanan layar }
  Sub XRead,Ax
@X1_No_Right_Crop:
  Mov Ax,Y1
  Cmp Ax,0
Jge @No_Top_Crop
  Neg Ax          { Kalkulasi Pembatasan atas layar }
  Mov YWStart,0
  Mov YRStart,Ax
  Sub YRead,Ax
@No_Top_Crop:
  Mov Ax,Y1
  Add Ax,Width
  Dec Ax
  Cmp Ax,199
Jle @No_Bot_Crop
  Sub Ax,199      { Kalkulasi Pembatasan bawah layar }
  Sub YRead,Ax
@No_Bot_Crop:
  Add di,XWStart { Perhitungan offset alamat alokasi pixel }
  Mov Ax,YWStart
  Shl Ax,2
  Add Ax,YWStart
  Shl Ax,6
  Add di,Ax

  Mov Ax,YRStart { Ax = Jumlah kolom sprite yang dilompati }
  Xor Ah,Ah       { Nol-kan Ah }
  Mul Length      { Kalikan Jumlah kolom yang di-skip dengan }
                  { jumlah baris }
  Add si,Ax       { Tambahkan pada SI }
  Add si,XRStart { Tambahkan start pembacaan secara horizontal }

  Mov Cx,YRead    { Set CX dengan total peletakan pixel secara }
                  { vertikal }
@Loop1:           { Vertikal Loop }
  Push Cx
```

```
    Push di
    Push si
    Mov Cx,XRead    { Set CX dengan total peletakan pixel }
                    { secara horizontal }
    @Loop2:         { Horizontal Loop }
       Lodsb        { Baca titik yang hendak diletakkan }
       or al,al     { Test apakah titik itu berwarna 0 }
       Jz @No_Write
       Stosb        { Kalau bukan 0 maka letakkan ke layar }
       dec di       { Kurangi DI }
     @No_Write:
       inc di       { Tambahkan DI }
    Loop @Loop2     { Loop ke Horizontal Loop sampai CX = 0 }
    Pop si          { Kembalikan SI ke harga sebelumnya }
    Pop di          { Kembalikan DI ke harga sebelumnya }
    Add si,Length   { Tambahkan SI dengan Length }
    Add di,320      { Tambahkan DI dengan 320 }
    Pop Cx          { Kembalikan counter CX untuk meng-counter }
                    { vertical loop }
    Loop @Loop1     { Loop ke Vertikal Loop }
  Pop ds            { Kembalikan stack }
End;
```

PENJELASAN PROGRAM

Program di atas adalah program yang akan menggantikan PUTIMAGE mulai saat ini. Memang cukup rumit, tapi sangatlah penting untuk mengerti bagaimana procedure di atas bekerja karena fungsi-fungsi yang akan kita bahas selanjutnya akan selalu menggunakan pendeteksian batas layar.

Sekarang kita akan membahas bagaimana cara kerja procedure di atas.

```
Procedure PutImage(Var PicBuff;X1,Y1:Integer;Var Buffer);Assembler;
Var XWStart:Word;
    XRStart:Word;   { Titik X Pertama yang dibaca }
    YWStart:Word;   { Koordinat Y Pertama untuk penulisan }
    YRStart:Word;   { Titik Y Pertama yang dibaca }
    Length:Word;    { Panjang sprite }
    Width:Word;     { Lebar sprite }
    XRead:Word;     { Jumlah peletakan titik secara horizontal }
    YRead:Word;     { Jumlah peletakan titik secara vertikal }
  Asm
```

XRStart dan YRStart adalah koordinat pertama pada sprite yang akan dibaca. Di sini sprite yang berukuran length x width dianggap sebagai sebuah screen.

YWStart dan XWStart adalah koordinat pertama peletakan sprite pada screen. XWStart dan YWStart ini tidak akan pernah melebihi batas layar.

Xread dan Yread adalah jumlah peletakan titik secara horizontal dan vertikal setelah sprite dipotong oleh batasan layar. Jika ada proses crop/sprite yang melebihi batas layar, Xread = Length dan Yread = Width.

```
Push ds             { Simpan ds ke stack }
Lds si,Buffer       { Alamatkan DS:SI ke Buffer }
Les di,PicBuff      { Alamatkan ES:DI ke PicBuff }
Mov Ax,[ds:si]      { Ambil panjang sprite }
Mov Length,Ax       { Simpan di Length }
```

Proses di atas sama dengan proses yang dilakukan pada PUTIMAGE yaitu proses untuk mengarahkan DS:SI dan ES:DI ke alamat memori. ES:DI adalah alamat tujuan = screen. DS:SI adalah alamat sprite.

```
Mov XRead,Ax        { Jumlah peletakan titik horizontal = Length }
```

Pertama-tama kita anggap sprite tidak melebihi batas samping layar sehingga Xread = length

```
Add si,2            { Arahkan SI ke lebar dari sprite }
Mov Ax,[ds:si]      { Ambil lebar sprite }
Mov Width,Ax        { Simpan di Width }
Mov YRead,Ax        { Jumlah peletakan titik vertikal = Width }
```

Kita anggap sprite tidak melebihi batas atas atau bawah layar sehingga Yread = width

```
Mov YRStart,0       { Asumsikan titik vertikal pertama yang }
                    { dibaca ialah titik pertama }
```

Kita asumsikan sprite tidak melebihi batas atas layar sehingga pembacaan vertikal tidak perlu melakukan lompatan. Jadi koordinat Y pada sprite yang akan dibaca pertama kali adalah Y = 0.

```
Mov Ax,Y1           { Ambil koordinat Y di mana sprite akan }
                    { diletakkan }
Mov YWStart,Ax      { Asumsikan koordinat sprite pada Y }
```

Kita asumsikan sprite tidak melebihi batas layar sehingga koordinat yang dikirim lewat parameter sudah valid. Maka koordinat YWStart = Y1

```
Add si,2          { Arahkan SI ke titik pertama dari sprite }
Mov XRStart,0     { Asumsikan titik horizontal pertama yang }
                  { dibaca ialah titik pertama }
Mov Ax,X1         { Ambil koordinat X di mana sprite akan }
                  { diletakkan }
Mov XWStart,Ax    { Asumsikan koordinat sprite pada X }
```

Kita asumsikan bahwa sprite tidak melebihi batas samping layar sehingga koordinat X yang dikirim lewat parameter sudah valid. Jadi XWStart = X1.

```
Cmp Ax,0          { Deteksi batas kiri layar }
```

Apakah koordinat X yang dikirim melebihi batas kiri layar? Jika koordinat yang dikirim negatif, sprite tersebut tentu saja melebihi batas kiri layar.

```
Jge @X1_Not_Negative
  Neg Ax          { Kalkulasi Pembatasan kiri layar }
  Mov XRStart,Ax
  Mov XWStart,0
  Sub XRead,Ax
```

Sprite menerobos batas kiri layar

Jika ternyata sprite tersebut melebihi batas kiri layar, koordinat negatif tersebut menunjukkan jumlah titik yang harus dilompati pada saat melakukan pembacaan secara horizontal. Berhubung koordinat tersebut negatif maka kita harus membuatnya positif. Kemudian Jumlah peletakan titik secara horizontal yaitu Xread dikurangi dengan jumlah titik yang dilompati.

```
@X1_Not_Negative:
  Mov Ax,X1
  Add Ax,Length
  Dec Ax
  Cmp Ax,319
```

Proses di atas ialah proses untuk mendeteksi apakah sprite melebihi batas kanan layar.

Jika Koordinat awal sprite ditambah dengan panjang dari sprite melebihi 319 maka sprite tersebut telah melebihi batas kanan layar karena koordinat

awal sprite + panjang sprite tersebut merupakan koordinat X dari sisi kanan sprite.

```
Jle @X1_No_Right_Crop
  Sub Ax,319              { Kalkulasi Pembatasan kanan layar }
  Sub XRead,Ax
```

Sprite menerobos sisi kanan layar

Proses untuk mengetahui jumlah titik di sebelah sisi kanan sprite yang dilompati adalah dengan mengurangi koordinat kanan sprite dengan 319. Kemudian jumlah peletakan titik Xread dikurangi dengan angka tersebut.

```
@X1_No_Right_Crop:
  Mov Ax,Y1
  Cmp Ax,0
```

Dilakukan proses untuk mendeteksi atas layar.

```
Jge @No_Top_Crop
  Neg Ax              { Kalkulasi Pembatasan atas layar }
  Mov YWStart,0
  Mov YRStart,Ax
  Sub YRead,Ax
```

Sprite menerobos atas layar

Jika sprite menerobos atas layar, dalam hal ini koordinat Y dari sprite negatif maka positifkan koordinat tersebut. Jumlah titik vertikal yang akan dilompati pada saat melakukan pembacaan sprite adalah koordinat Y yang dipositifkan tersebut. Oleh karena itu Yread dikurangi.

```
@No_Top_Crop:
  Mov Ax,Y1
  Add Ax,Width
  Dec Ax
  Cmp Ax,199
```

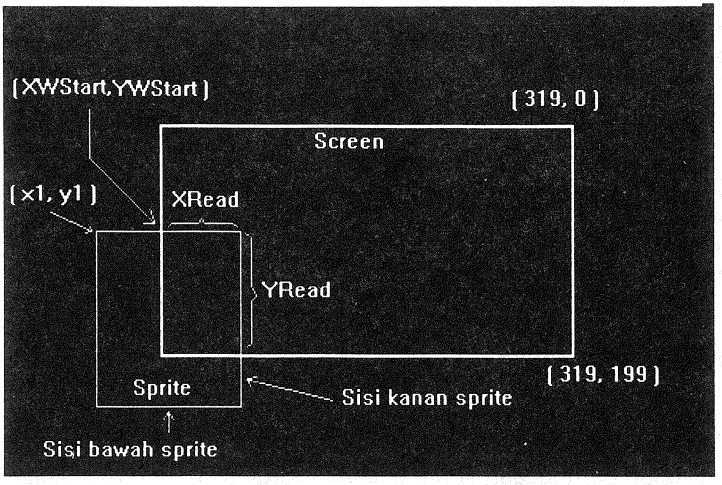
Deteksi apakah sprite menerobos bawah layar. Caranya ialah dengan menambahkan koordinat Y dari sprite dengan Lebar dari sprite. Hasil perhitungan ini menunjukkan koordinat sisi bawah dari sprite. Jika ternyata melebihi 199 maka sprite menerobos bawah layar.

```
Jle @No_Bot_Crop
  Sub Ax,199          { Kalkulasi Pembatasan bawah layar }
  Sub YRead,Ax
```

Sprite menerobos batas bawah layar

Kurangi koordinat sisi bawah sprite dengan 199. Angka ini menunjukkan jumlah titik pada sisi bawah sprite yang harus dilompati/tidak ditampilkan. Sehingga Yread dikurangi dengan angka tersebut.

Untuk membantu Anda dalam mempelajari teknik pembatasan di atas, lihatlah diagram berikut.



Gambar di atas melukiskan sebuah sprite yang menerobos batas kiri dan bawah layar. Tentu saja Xread dan Yread yang akan dipergunakan dalam peletakan titik tidak sama dengan length dan width dari sprite, karena Xread dan Yread tersebut telah dikurangi oleh titik yang melebihi batas layar. Perhatikan pula XWStart dan YWStart, keduanya tetap berada pada koordinat (0,0) bukannya pada koordinat (X1,Y1) karena telah dilakukan proses pembatasan.

Selanjutnya lihatlah diagram berikut.

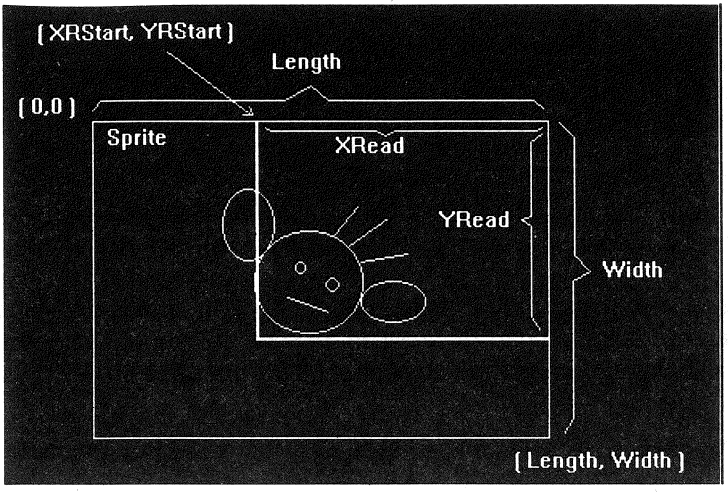


Diagram ini menunjukkan situasi sprite yang akan ditampilkan pada diagram yang lalu.

Cobalah lihat kembali diagram sebelumnya di mana sprite akan menerobos batas kiri dan bawah layar. Sehingga sprite tersebut akan terpotong bagian kiri dan bawahnya, karena bagian tersebut berada di luar layar. Pada situasi ini Xread dan Yread tidak sama dengan Length dan Width. XRStart dan Ystart pada diagram di atas menunjukkan titik pertama yang akan diletakkan pada layar. Lihatlah diagram di atas dan juga diagram sebelumnya, XRStart bisa didapat secara mudah dengan cara mengambil harga koordinat X1 yang dipositifkan karena koordinat (X1,Y1) pada layar sama dengan koordinat sprite (0,0)

```
@No_Bot_Crop:
  Add di,XWStart      { Perhitungan offset alamat alokasi pixel }
  Mov Ax,YWStart
  Shl Ax,2
  Add Ax,YWStart
  Shl Ax,6
  Add di,Ax
```

Proses di atas ini sebenarnya tidak berbeda dengan proses pada PUTIMAGE kita yang lama, yaitu proses untuk menentukan alamat memori yang sebenarnya/proses perhitungan offset memori dari koordinat X dan Y. Dalam hal ini X = XWStart dan Y = YWStart. Kedua variabel ini menunjukkan start pertama kita akan melakukan peletakan titik. Ingat bahwa titik pertama tersebut tidak selalu titik pertama pada sprite yang akan kita letakkan (seperti halnya pada PUTIMAGE yang lama). Titik pertama ini ialah titik pada koordinat sprite (XRStart, YRStart).

```
Mov Ax,YRStart   { Ax = Jumlah kolom sprite yang dilompati }
Xor Ah,Ah        { Nol- kan Ah }
Mul Length       { Kalikan Jumlah kolom yang di-skip }
                 { dengan jumlah baris }
Add si,Ax        { Tambahkan pada SI }
Add si,XRStart   { Tambahkan start pembacaan secara }
                 { horizontal }
```

Proses di atas adalah proses untuk menambah SI dengan jumlah titik yang dilompati. Caranya ialah dengan mengalikan YRStart dengan panjang dari gambar. Kemudian tambahkan hasil perhitungan tersebut pada SI.

```
Mov Cx,YRead     { Set CX dengan total peletakan pixel secara }
                 { vertikal }
```

Set Counter pada Vertikal loop = Yread

Dengan ini maka program akan melakukan peletakan titik vertikal sebanyak Yread.

```
@Loop1:          { Vertikal Loop }
    Push Cx
    Push di
    Push si
    Mov Cx,XRead { Set CX dengan total peletakan pixel secara }
                 { horizontal }
```

Simpan CX yang digunakan untuk counter vertikal loop.

Simpan juga DI dan SI.

```
@Loop2:          { Horizontal Loop }
   Lodsb         { Baca titik yang hendak diletakkan }
   or al,al      { Test apakah titik itu berwarna 0 }
   Jz @No_Write
   Stosb         { Kalau bukan 0 maka letakkan ke layar }
   dec di        { Kurangi DI }
 @No_Write:
   inc di        { Tambahkan DI }
Loop @Loop2      { Loop ke Horizontal Loop sampai CX = 0 }
```

Proses di atas sama persis dengan proses PUTIMAGE kita yang lama, yaitu membaca sebuat titik dari sprite kemudian jika titik tidak berwarna 0, letakkan titik tersebut pada screen. Proses ini dilakukan secara horizontal loop.

```
Pop si              { Kembalikan SI ke harga sebelumnya }
Pop di              { Kembalikan DI ke harga sebelumnya }
Add si,Length       { Tambahkan SI dengan Length }
Add di,320          { Tambahkan DI dengan 320 }
```

Kembalikan SI dan DI. Kemudian arahkan SI ke koordinat sprite Y berikutnya yaitu dengan menambahkannya dengan Length. Tambahkan DI ke koordinat screen Y berikutnya, yaitu dengan menambahnya 320.

```
Pop Cx          { Kembalikan counter CX untuk meng-counter  }
                { vertikal loop }
Loop @Loop1     { Loop ke Vertikal Loop }
```

Kembalikan counter CX yang akan kita gunakan untuk counter vertikal loop.

```
Pop ds          { Kembalikan stack }
```

Kembalikan ds dari stack.

```
End;
```

Phew!!! Cukup rumit bukan. Saya harap dengan penjelasan di atas telah cukup membantu Anda untuk dapat menguasai teknik ini.

Nah sekarang kita akan mencoba PUTIMAGE kita yang baru dengan program di bawah ini.

```
{-----------------------------------------------}
{ Program: B_PUT1.PAS                           }
{ Coder : Agustinus Nalwan                      }
{ Fungsi: Mencoba PutImage dengan batasan layar }
{-----------------------------------------------}

Uses Crt,Dos,VgaCad;

Var Pic     : Pointer;
    Process: Pointer;
    Back    : Pointer;
    Finish : Boolean;
    X,Y     : Integer;
    Ch      : Char;
    Crop    : Boolean;
Begin
  OpenVga256;
  GetMem(Pic,30000);
  GetMem(Process,64000);
```

```
  GetMem(Back,64000);
  Install_Palette('Palette.Pal');
  Load_Picture('Pics1.',Screen^);
  GetImage(Screen^,1,1,108,86,Pic^);
  Load_Picture('Pics2.',Back^);
  X:=50;Y:=50;
  Finish:=False;
  Crop:=True;
  Repeat
    Repeat
      MoveTo(Back^,Process^);
      If Crop then PutImage(Process^,X,Y,Pic^) else
        PutOld(Process^,X,Y,Pic^);
      MoveTo(Process^,Screen^);
    Until Keypressed;
    Ch:=Readkey;
    Case Ch of
      #80:Inc(Y,3);
      #72:Dec(Y,3);
      #75:Dec(X,3);
      #77:Inc(X,3);
      #27:Finish:=True;
      ' ':Crop:=Not Crop;
    End;
  Until Finish;
  Readln;
End.
```

PENJELASAN PROGRAM

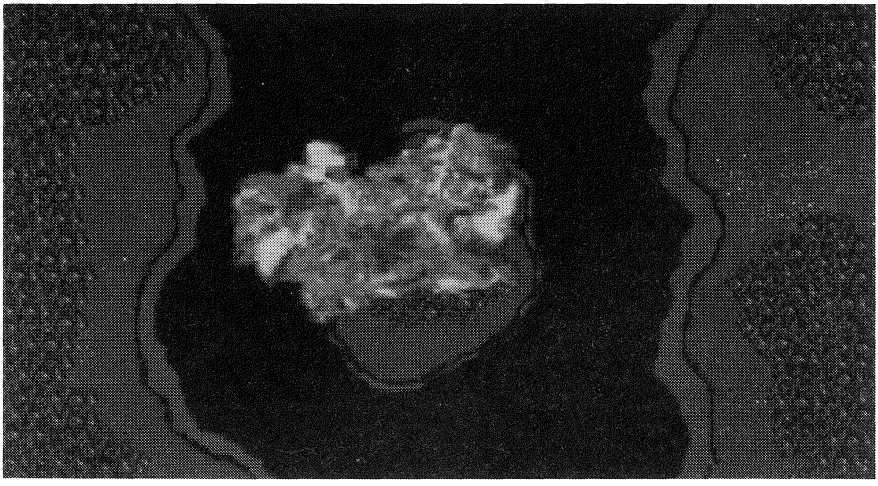
Mengingat kita sekarang telah mempelajari tahap kedua GAME PROGRAMMING, saya tidak akan menjelaskan secara detail cara kerja program di atas. Pembaca diharapkan telah menguasai teknik pemrograman sederhana seperti pada program B_Put1.PAS di atas, mengingat hal tersebut telah dijelaskan pada buku seri pertama SAP PEMROGRAMAN GAME DAN ANIMASI PROFESIONAL.

Cobalah jalankan program di atas, Anda akan melihat sebuah gambar awan yang berada di atas hutan. Cobalah gerakkan awan tersebut hingga menerobos batas layar. Pada saat menerobos batas layar, sprite dipotong dengan tepatnya. Ingat PutImage dengan batasan layar ini tidak dirancang untuk situasi seluruh bagian sprite menerobos batas layar. Jadi paling tidak harus ada bagian sprite yang terlihat.

Tombol spacebar dapat Anda gunakan untuk men-switch PUTIMAGE menjadi PUTIMAGE yang lama yang masih belum menggunakan pembatasan layar. PUTIMAGE yang lama ini saya ganti namanya menjadi PUTOLD. Co-

balah tabrak batasan layar menggunakan mode PUTOLD, potongan gambar akan muncul di sisi yang berlawanan.

Untuk keluar dari program, Anda tinggal menekan tombol ESC kemudian ENTER.



Gambar 1.1 Output dari B_PUT1.PAS

PELETAKAN SPRITE SATU WARNA

Setelah kita mempelajari bagaimana cara kerja PUTIMAGE dengan batasan layar, sekarang kita akan beranjak ke tingkat yang lebih tinggi lagi. Pernahkah Anda melihat game ZONE 66?

Game ini adalah game bertipe pesawat tempur yang menggunakan sistem parallax scrolling/scroll segala arah. Jika Anda menekan tombol turbo, pesawat Anda akan terbang dengan kecepatan yang cepat sekali. Sehingga pada tempat di mana Anda berada sebelumnya akan membekas gambar-gambar pesawat Anda dengan warna hitam. Dalam hal ini kita tidak perlu menciptakan sprite pesawat kita dengan warna yang berbeda-beda. Hal ini akan membuang memori banyak.

Inilah yang disebut dengan SPRITE SATU WARNA. Logika peletakan titik dari sprite ini adalah sebagai berikut; Jika titik tersebut berwarna 0, jangan diletakkan tetapi jika bukan 0 maka letakkan titik dengan warna yang diinputkan pada parameter. Jadi sprite tersebut nantinya akan muncul dengan satu warna saja.

Procedure ini adalah merupakan fungsi tambahan bagi VGACAD.TPU.

```
Procedure Put1Col(Var PicBuff;X1,Y1:Integer;Col:Byte;
Var Buffer);Assembler;
Var XWStart:Word;
    XRStart:Word;
    YWStart:Word;
    YRStart:Word;
    Length:Word;
    Width:Word;
    XRead:Word;
    YRead:Word;
  Asm
    Push ds
    Lds si,Buffer
    Les di,PicBuff
    Mov Ax,[ds:si]
    Mov Length,Ax
    Mov XRead,Ax
    Add si,2
    Mov Ax,[ds:si]
    Mov Width,Ax
    Mov YRead,Ax
    Mov YRStart,0
    Mov Ax,Y1
    Mov YWStart,Ax
    Add si,2
    Mov XRStart,0
    Mov Ax,X1
    Mov XWStart,Ax
    Cmp Ax,0
    Jge @X1_Not_Negative
      Neg Ax
      Mov XRStart,Ax
      Mov XWStart,0
      Sub XRead,Ax
    @X1_Not_Negative:
      Mov Ax,X1
      Add Ax,Length
      Dec Ax
      Cmp Ax,319
    Jle @X1_No_Right_Crop
      Sub Ax,319
      Sub XRead,Ax
    @X1_No_Right_Crop:
```

```
    Mov Ax,Y1
    Cmp Ax,0
  Jge @No_Top_Crop
    Neg Ax
    Mov YWStart,0
    Mov YRStart,Ax
    Sub YRead,Ax
  @No_Top_Crop:
    Mov Ax,Y1
    Add Ax,Width
    Dec Ax
    Cmp Ax,199
  Jle @No_Bot_Crop
    Sub Ax,199
    Sub YRead,Ax
  @No_Bot_Crop:
    Add di,XWStart
    Mov Ax,YWStart
    Shl Ax,2
    Add Ax,YWStart
    Shl Ax,6
    Add di,Ax

    Mov Ax,YRStart
    Xor Ah,Ah
    Mul Length
    Add si,Ax
    Add si,XRStart

    Mov Cx,YRead
    Mov Bl,Col
  @Loop1:
    Push Cx
    Push di
    Push si
    Mov Cx,XRead
    @Loop2:
        Lodsb
        or al,al
        Jz @No_Write
        Mov [Es:Di],Bl
        @No_Write:
        inc di
    Loop @Loop2
    Pop si
    Pop di
    Add si,Length
    Add di,320
    Pop Cx
    Loop @Loop1
  Pop ds
End;
```

PENJELASAN PROGRAM

Karena procedure ini juga menggunakan sistem pembatasan layar, maka cara kerja dari procedure ini sama persis dengan cara kerja procedure PUTIMAGE di atas, hanya bedanya sewaktu dilakukan peletakan titik, kita tidak meletakkan titik yang kita baca dari sprite melainkan meletakkan titik yang diberikan pada parameter.

```
@Loop2:
   Lodsb
   or al,al
   Jz @No_Write
   Mov [Es:Di],Bl
 @No_Write:
   inc di
Loop @Loop2
```

Ini adalah bagian horizontal loop pada procedure PUT1COL. Di sini dapat Anda lihat bahwa tidak dilakukan proses STOSB melainkan proses MOV dari Bl yang berisikan parameter COL.

Register Bl ini diisikan dengan parameter COL pada saat sebelum dilakukannya vertikal loop (lihatlah listing program).

Nah sekarang ayo kita buat sebuah program yang menggunakan PUT1COL ini.

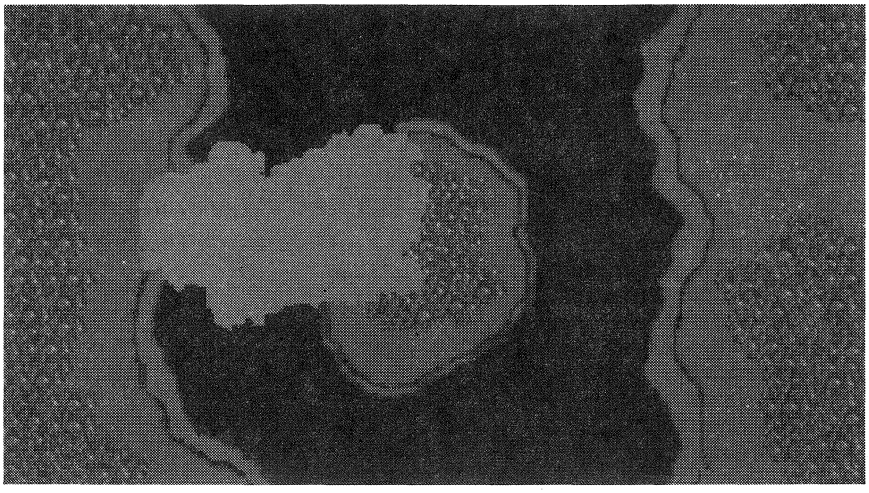
```
{-------------------------------------}
{ Program: B_PUT2.PAS                 }
{ Coder : Agustinus Nalwan            }
{ Fungsi: Mencoba Put satu warna      }
{-------------------------------------}

Uses Crt,Dos,VgaCad;

Var Pic     : Pointer;
    Process: Pointer;
    Back    : Pointer;
    Finish : Boolean;
    X,Y     : Integer;
    Ch      : Char;
    Col1    : Boolean;
Begin
  OpenVga256;
  GetMem(Pic,30000);
  GetMem(Process,64000);
  GetMem(Back,64000);
  Install_Palette('Palette.Pal');
```

```
  Load_Picture('Pics1.',Screen^);
  GetImage(Screen^,1,1,108,86,Pic^);
  Load_Picture('Pics2.',Back^);
  X:=50;Y:=50;
  Finish:=False;
  Col1:=True;
  Repeat
    Repeat
      MoveTo(Back^,Process^);
      If Col1 then
        PutImage(Process^,X,Y,Pic^) else
        Put1Col(Process^,X,Y,12,Pic^);
      MoveTo(Process^,Screen^);
    Until Keypressed;
    Ch:=Readkey;
    Case Ch of
      #80:Inc(Y,3);
      #72:Dec(Y,3);
      #75:Dec(X,3);
      #77:Inc(X,3);
      #27:Finish:=True;
      ' ':Col1:=Not Col1;
    End;
  Until Finish;
  Readln;
End.
```

PENJELASAN PROGRAM

Progam di atas sama persis dengan program yang kita gunakan untuk mencoba PUTIMAGE kita yang baru. Hanya di sini kita gunakan PUTIMAGE dan PUT1COL dalam meletakkan sprite. Dengan begitu kita dapat melihat perbedaan sprite yang diletakkan melalui PUTIMAGE dan melalui PUT1COL.

PUT1COL ini memang mudah dibuatnya, tetapi nantinya akan memegang peranan penting dalam pembuatan game-game profesional. Biasanya teknik ini digunakan untuk memberikan efek suatu benda yang bergerak dengan kecepatan tinggi. Juga kadang digunakan dalam ledakan-ledakan atau percikan api. Hal-hal seperti ini akan kita pelajari pada bab-bab berikutnya.



Gambar 1.2 Output dari B_PUT2.PAS

TEKNIK PELETAKAN SPRITE TRANSPARANT

Teknik transparant ini baru dipergunakan pada sekitar awal tahun 1995 ini. Teknik ini memungkinkan kita untuk meletakkan sebuah sprite, tetapi gambar di bawah sprite tersebut masih kelihatan samar-samar. Kalau Anda pernah mencoba game HERETIC Anda akan melihat teknik ini. Pada HERETIC, bila Anda mengambil BLUR SPHERE, Anda akan menjadi transparant sehingga sulit dilihat oleh musuh.

Adapun pembuatan efek ini ialah dengan menggunakan sistem hash table.

Caranya: setiap kali Anda akan meletakkan suatu titik pada screen, program akan membaca pula titik (pada screen) yang akan ditumpuk dengan titik baru tersebut. Kedua titik ini digunakan sebagai index array 2 dimensi yang berisi daftar informasi hasil penggabungan dua buah warna. Nah, warna titik yang didapat dari 2 index tersebut adalah titik yang akan diletakkan pada screen.

Karena warna 0 tidak akan diproses oleh PUTIMAGE maka kita memerlukan array 2 dimensi sebesar 256 x 255 = 65280 bytes. Memang teknik ini amat memboroskan memori, tapi hash table ini terkenal dengan kecepatannya

sehingga kita akan tetap memakai teknik ini untuk menambah fasilitas VGACAD.TPU.

Misalnya kita hendak meletakkan warna nomor 13 pada koordinat (24, 26), program akan membaca titik pada koordinat (24,26). Misalnya titik pada koordinat tersebut adalah titik berwarna nomor 28 maka selanjutnya program akan mengambil warna dari TABEL[13,28] sebagai titik yang merupakan gabungan dari kedua titik tersebut. Titik inilah yang akan dialokasikan ke screen.

Dengan kata lain, di dalam hash tabel ini diisikan nomor warna yang merupakan hasil pencampuran dari tiap pasangan warna.

Nah sekarang bagaimana cara pembuatan hash tabel tersebut? Biasanya tabel ini dibuat terlebih dahulu oleh program lain dan kemudian di simpan ke dalam sebuah file. Dengan begitu setiap kita menjalankan program animasi kita, kita tidak perlu menciptakan tabel tersebut, melainkan tinggal membacanya dari file.

Teknik pembuatan tabel tersebut didasari oleh logika sebagai berikut.

Karena tabel ini merupakan array 2 dimensi, maka setiap cell dari tabel ini akan memiliki dua buah index. Kita gunakan dua buah index ini sebagai warna yang akan dicampurkan. Nah sekarang masalahnya bagaimana mencari warna campuran dari dua buah warna?

Jika dua buah warna dicampurkan, warna hasil campurannya ialah hasil rata-rata dari ketiga intensitas warna yang dimiliki oleh dua warna tersebut. Misalnya warna pertama memiliki intensitas RGB = 10 20 30, sedangkan warna kedua memiliki intensitas RGB = 30 50 40. Maka

warna hasil campuran keduanya ialah warna dengan intensitas RGB (10+30)/2 (20+50)/2 (30+40)/2 yaitu warna dengan intensitas RGB 20 35 35.

Yang perlu diingat ialah warna dengan intensitas RGB 20 35 35 itu belum tentu ada di dalam palette warna yang sedang Anda gunakan. Sebenarnya Anda bisa saja mengubah salah satu warna dari 256 warna yang ada menjadi warna dengan intensitas tersebut dengan menggunakan SETPALETTE. Tetapi hal ini tidak dianjurkan, karena selain lambat gambar yang menggunakan warna tersebut akan ikut berubah sehingga tampilan Anda akan menjadi berantakan. Tentunya Anda dapat membayangkan kalau setiap saat program mengubah satu intensitas warna.

Nah pemecahannya adalah sebagai berikut. Setelah Anda mendapat warna hasil campuran tersebut, cari salah satu warna pada susunan palette yang Anda gunakan yang paling mendekati warna tersebut. Dengan begini kita tidak perlu mengubah intensitas suatu warna.

Adapun cara mencari warna yang paling mendekati suatu warna adalah sebagai berikut.

Jika warna hasil campuran tersebut memiliki intensitas RGB 20 35 35, Anda tinggal menelusuri seluruh nomor warna pada 256 susunan palette dan melakukan proses pembedaan pada setiap nomor warna. Proses pembedaan yang dilakukan ialah dengan melakukan perhitungan:

$$DMin=Abs(R1-R2)+Abs(G1-G2)+Abs(B1-B2)$$

R1 G1 B1 adalah intensitas warna campuran.

R2 G2 B2 adalah intensitas warna yang sedang Anda lakukan proses pembedaan.

Nomor warna yang memiliki DMin paling kecil adalah warna yang terdekat dengan warna campuran tersebut. Untuk mencari DMin yang terkecil Anda dapat menggunakan sorting, yang paling efisien ialah selection sort.

Untuk lebih jelasnya lihatlah program di bawah ini.

```
Procedure Search_Transparent_Palette(PalTable:Pointer;Name:String);
Var A,B,Col1,
    Col2              : Integer;
    Red1,Red2         : Integer;
    Green1,Green2     : Integer;
    Blue1,Blue2       : Integer;
    SRed,SGreen,
    SBlue             : Integer;

    DRed              : Integer;
    DGreen            : Integer;
    DBlue             : Integer;
    Red,Green,
    Blue              : Integer;
    DTotal            : Integer;
    Dmin              : Integer;
    TransCol          : Byte;
    FileTable         : File;
    Ch                : Char;
  Begin
    Assign(FileTable,Name);
    Rewrite(FileTable,1);
    For Col1:=1 to 255 do
      Begin
        Red1:=Default_Pal^[Col1].R;
        Green1:=Default_Pal^[Col1].G;
        Blue1:=Default_Pal^[Col1].B;
```

```
      For Col2:=0 to 255 do
        Begin
          Red2:=Default_Pal^[Col2].R;
          Green2:=Default_Pal^[Col2].G;
          Blue2:=Default_Pal^[Col2].B;

          SRed:=(Red1+Red2)      div 2;
          SGreen:=(Green1+Green2) div 2;
          SBlue:=(Blue1+Blue2)    div 2;
          DMin:=1000;
          For A:=0 to 255 do
            Begin
              Red:=Default_Pal^[A].R;
              Green:=Default_Pal^[A].G;
              Blue:=Default_Pal^[A].B;

              DRed :=Abs(Red-SRed);
              DGreen:=Abs(Green-SGreen);
              DBlue:=Abs(Blue-SBlue);
              DTotal:=DRed+DGreen+DBlue;
              If (DTotal<DMin) or (DTotal = 0) then
                Begin
                  DMin:=DTotal;
                  TransCol:=A;
                End;
            End;
          BlockWrite(FileTable,TransCol,1);
        End;
      End;
    Close(FileTable);
  End;
```

PENJELASAN PROGRAM

Procedure di atas adalah fasilitas tambahan dari VGACAD.TPU yang juga bertujuan untuk membentuk palette tabel kemudian menyimpannya dalam sebuah file. Procedure ini keseluruhannya dibentuk dengan menggunakan Turbo Pascal routine, karena procedure ini tidak akan kita gunakan di tengah-tengah looping pada program-program yang akan kita buat. Jadi masalah kecepatan tidak perlu diperhatikan di sini.

```
Procedure Search_Transparent_Palette(PalTable:Pointer;Name:String);
Var A,B,Col1,
    Col2              : Integer;
    Red1,Red2         : Integer;
    Green1,Green2     : Integer;
    Blue1,Blue2       : Integer;
    SRed,SGreen,
    SBlue             : Integer;
```

```
    DRed            : Integer;
    DGreen          : Integer;
    DBlue           : Integer;
    Red,Green,
    Blue            : Integer;
    DTotal          : Integer;
    DMin            : Integer;
    TransCol        : Byte;
    FileTable       : File;
    Ch              : Char;
  Begin
    Assign(FileTable,Name);
    Rewrite(FileTable,1);
```

Buka file yang akan kita gunakan untuk menyimpan tabel transparant.

```
    For Col1:=1 to 255 do
      Begin
```

Dilakukan loop mulai warna 1 hingga 255. Warna 0 tidak perlu kita proses, karena warna ini tidak akan diproses pada saat kita meletakkan Sprite.

```
        Red1:=Default_Pal^[Col1].R;
        Green1:=Default_Pal^[Col1].G;
        Blue1:=Default_Pal^[Col1].B;
```

Ambil intensitas RGB dari warna pertama.

```
        For Col2:=0 to 255 do
         Begin
```

Looping untuk warna kedua mulai 0 hingga 255. Mengingat warna kedua ini nantinya di dapat dari screen yang ditumpuk oleh warna pertama, maka ada kemungkinan di screen tersebut terdapat warna 0. Untuk itu kita perlu memproses warna 0 juga.

```
        Red2:=Default_Pal^[Col2].R;
        Green2:=Default_Pal^[Col2].G;
        Blue2:=Default_Pal^[Col2].B;
```

Ambil intensitas RGB dari warna kedua.

```
SRed:=(Red1+Red2)      div 2;
SGreen:=(Green1+Green2) div 2;
SBlue:=(Blue1+Blue2)    div 2;
```

Hitung warna campuran dari warna pertama dan warna kedua. Sred Sgreen Sblue adalah intensitas campuran dari kedua warna tersebut. Proses selanjutnya ialah mencari warna terdekat yang memiliki intensitas RGB Sred Sgreen Sblue.

```
Dmin:=1000;
```

Set Dmin ke harga yang besar sekali.

```
For A:=0 to 255 do
  Begin
```

Lakukan loop mulai warna nomor 0 sampai 255 untuk mencari warna terdekat.

```
Red:=Default_Pal^[A].R;
Green:=Default_Pal^[A].G;
Blue:=Default_Pal^[A].B;

DRed :=Abs(Red-SRed);
DGreen:=Abs(Green-SGreen);
DBlue:=Abs(Blue-SBlue);
DTotal:=DRed+DGreen+DBlue;
```

Ambil Intensitas warna tersebut untuk melakukan proses pembedaan. Kemudian dilakukan proses menghitung Dtotal. Dtotal ini menunjukkan radius perbedaan warna tersebut dengan warna campuran.

Kita harus mencari warna dengan radius perbedaan yang terkecil.

```
If (DTotal<DMin) or (DTotal = 0) then
  Begin
    DMin:=DTotal;
    TransCol:=A;
```

Jika Ditemukan warna yang memiliki radius perbedaan lebih sedikit dari Dmin, ambil warna tersebut dan masukkan ke TransCol. Dmin diset ke radius perbedaan dari warna tersebut.

Dengan cara ini pada akhir loop, warna yang memiliki radius perbedaan terkecil dapat Anda lihat di variabel TransCol.

```
            End;
          End;
        BlockWrite(FileTable,TransCol,1);
```

Simpan warna yang ditemukan dalam tabel transparant. Penulisan tabel ini dalam file dilakukan secara beranting. Besar dari file ini adalah 255*256 = 65280 bytes.

```
        End;
      End;
    Close(FileTable);
```

Tutup file tabel transparant.

```
  End;
```

Sekarang kita akan mempelajari bagaimana memanfaatkan procedure ini untuk membentuk sebuah tabel transparant dari suatu susunan palette. Ingat lho! Tabel transparant ini tidak dapat Anda gunakan pada sembarang susunan palette, karena tabel itu hanya kompatibel dengan susunan palette yang Anda gunakan pada saat dibentuk. Jadi susunan palette yang berbeda memiliki tabel transparant yang berbeda.

Nah coba jalankan program di bawah ini. Program ini akan membentuk sebuah tabel transparant.

```
{-------------------------------------------}
{ Program: B_TR1.PAS                        }
{ Coder : Agustinus Nalwan                  }
{ Fungsi: Membentuk sebuah tabel transparant }
{-------------------------------------------}

Uses Crt,Dos,VgaCad;

Begin
  OpenVga256;
  Install_Palette('Palette.Pal');
  Search_Transparent_Palette('Trans.TAB');
End.
```

PENJELASAN PROGRAM

Programmnya sangat sederhana bukan?

Apabila program ini Anda jalankan, ia akan membuat sebuah file yaitu TRANS.TAB yang besarnya 65280 bytes. Ingat bahwa tabel transparant ini hanya berlaku jika Anda sedang menggunakan palette yang disimpan di

dalam file PALETTE.PAL. Karena tabel transparant ini Anda buat pada saat Anda menggunakan susunan palette dari PALETTE.PAL.

Setelah tabel transparant terbentuk, kita sekarang akan mencoba membuat sebuah program yang akan memanfaatkan tabel transparant tersebut.

Cobalah lihat program di bawah ini.

```
{--------------------------------------}
{ Program: B_TR2.PAS                   }
{ Coder : Agustinus Nalwan             }
{ Fungsi: Mencoba Put Transparant      }
{--------------------------------------}

Uses Crt,Dos,VgaCad;

Var Pic     : Pointer;
    PalTable: Pointer;
    Process: Pointer;
    Back    : Pointer;
    Finish : Boolean;
    X,Y     : Integer;
    Ch      : Char;
    Trans   : Boolean;
Begin
  OpenVga256;
  GetMem(Pic,30000);
  GetMem(PalTable,65280);
  GetMem(Process,64000);
  GetMem(Back,64000);
  Install_Palette('Palette.Pal');
  Install_Transparent(PalTable,'Trans.Tab');
  Load_Picture('Pics1.',Screen^);
  GetImage(Screen^,1,1,108,86,Pic^);
  Load_Picture('Pics2.',Back^);
  X:=50;Y:=50;
  Trans:=False;
  Finish:=False;
  Repeat
    Repeat
      MoveTo(Back^,Process^);
      If Trans then
        T_PutImage(Process^,X,Y,Pic^,PalTable^) else
          PutImage(Process^,X,Y,Pic^);
      MoveTo(Process^,Screen^);
    Until Keypressed;
    Ch:=Readkey;
    Case Ch of
      #80:Inc(Y,3);
      #72:Dec(Y,3);
      #75:Dec(X,3);
      #77:Inc(X,3);
      ' ':Trans:=Not Trans;
```

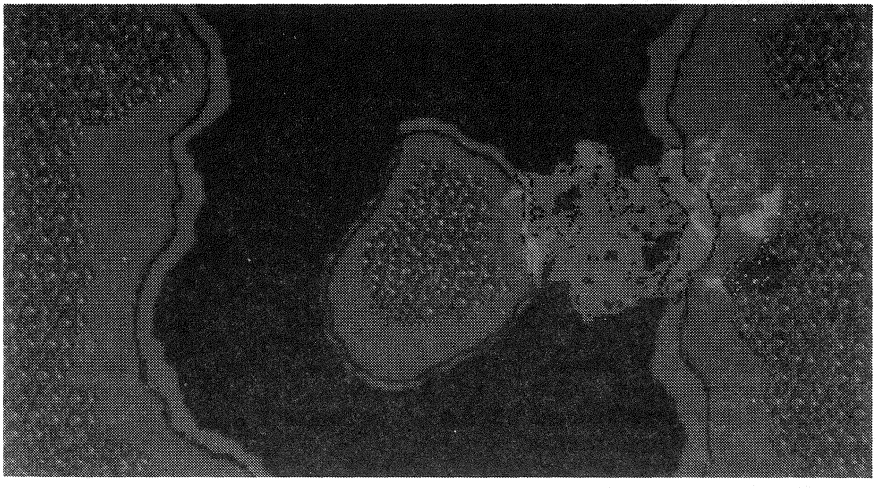
```
      #27:Finish:=True;
    End;
  Until Finish;
  Readln;
End.
```

PENJELASAN PROGRAM

Cobalah jalankan program tersebut.

Anda akan mendapati gambar awan di tengah-tengah hutan (awan lagi-awan lagi, bosen nih). Coba tekan spacebar, tombol ini digunakan untuk men-switch peletakan sprite menjadi transparant atau biasa. Peletakan transparant ini menggunakan sistem pembatasan layar juga lho! Dan satu lagi dibuatnya memakai bahasa Assembler juga. Pada buku ini kita akan lebih banyak bermain-main dengan bahasa Assembler.

Jika Anda perhatikan, program ini memiliki struktur yang hampir sama pula dengan program sebelumnya. Hanya ada beberapa perbedaan yaitu pada awal program dilakukan instalasi tabel palette.



Gambar 1.2b Output dari B_TR2.PAS

```
Begin
  OpenVga256;
  GetMem(Pic,30000);
  GetMem(PalTable,65280);
  GetMem(Process,64000);
  GetMem(Back,64000);
  Install_Palette('Palette.Pal');
  Install_Transparent(PalTable,'Trans.Tab');
```

Procedure Install_Transparent adalah fungsi tambahan dari VGACAD.TPU yang bertujuan untuk membaca tabel transparant dari suatu file ke dalam Alamat memori sebesar 65280 yang akan kita gunakan sebagai hash table dalam program kita.

Sebenarnya apa sih yang dilakukan oleh procedure ini? Lihatlah listing program dari procedure Install_Tansparent di bawah ini.

```
Procedure Install_Transparent(PalTable:Pointer;Name:String);
Var FileTable:File;
  Begin
    Assign(FileTable,Name);
    Reset(FileTable,1);
    BlockRead(FileTable,PalTable^,65280);
    Close(FileTable);
  End;
```

Seperti yang Anda lihat sendiri, procedure ini hanya membaca file transparant ke dalam memori yang disediakan untuk tabel transparant. Tentu saja Anda tidak mengalami kesulitan dalam mempelajari procedure ini kan?

Nah sekarang kita akan melihat apa yang terjadi di dalam procedure T_PutImage yang juga merupakan fungsi tambahan bagi VGACAD.TPU.

```
Procedure T_PutImage(Var PicBuff;X1,Y1:Integer;Var Buffer;
Var PalTable);Assembler;
Var XWStart:Word;
    XRStart:Word;
    YWStart:Word;
    YRStart:Word;
    Length:Word;
    Width:Word;
    XRead:Word;
    YRead:Word;
  Asm
    Push ds
    Lds si,Buffer
    Les di,PicBuff
    Mov Ax,[ds:si]
    Mov Length,Ax
    Mov XRead,Ax
```

```
Add si,2
Mov Ax,[ds:si]
Mov Width,Ax
Mov YRead,Ax
Mov YRStart,0
Mov Ax,Y1
Mov YWStart,Ax
Add si,2
Mov XRStart,0
Mov Ax,X1
Mov XWStart,Ax
Cmp Ax,0
Jge @X1_Not_Negative
  Neg Ax
  Mov XRStart,Ax
  Mov XWStart,0
  Sub XRead,Ax
@X1_Not_Negative:
  Mov Ax,X1
  Add Ax,Length
  Dec Ax
  Cmp Ax,319
Jle @X1_No_Right_Crop
  Sub Ax,319
  Sub XRead,Ax
@X1_No_Right_Crop:
  Mov Ax,Y1
  Cmp Ax,0
Jge @No_Top_Crop
  Neg Ax
  Mov YWStart,0
  Mov YRStart,Ax
  Sub YRead,Ax
@No_Top_Crop:
  Mov Ax,Y1
  Add Ax,Width
  Dec Ax
  Cmp Ax,199
Jle @No_Bot_Crop
  Sub Ax,199
  Sub YRead,Ax
@No_Bot_Crop:
  Add di,XWStart
  Mov Ax,YWStart
  Shl Ax,2
  Add Ax,YWStart
  Shl Ax,6
  Add di,Ax

  Mov Ax,YRStart
  Xor Ah,Ah
  Mul Length
  Add si,Ax
  Add si,XRStart
```

```
    Mov Cx,YRead
  @Loop1:
    Push Cx
    Push di
    Push si
    Mov Cx,XRead
    @Loop2:
        Lodsb
        or al,al
        Jz @No_Write

        Push bx
        Mov bl,[es:di]
        Xor bh,bh
        Xor ah,ah
        Dec al
        Shl ax,8
        Add ax,bx
        Push es
        Push di
        Les di,PalTable
        Add di,ax
        Mov al,[es:di]
        Pop di
        Pop es
        Pop bx

        Stosb
        dec di
      @No_Write:
        inc di
    Loop @Loop2
    Pop si
    Pop di
    Add si,Length
    Add di,320
    Pop Cx
    Loop @Loop1
  Pop ds
End;
```

PENJELASAN PROGRAM

Sebenarnya bagian awal dari procedure T_PUTIMAGE ini hampir sama dengan PUTIMAGE biasa, hanya pada saat akan dilakukannya peletakan titik di screen dilakukan proses yang berbeda. Yaitu dilakukannya pembacaan titik di screen pada koordinat di mana kita hendak meletakkan titik untuk menentukan index yang kedua. Setelah itu dua warna ini (satu adalah warna titik yang akan diletakkan, satu lagi adalah warna titik dari layar yang akan

ditumpuk oleh titik pertama) digunakan sebagai index tabel transparant untuk menentukan hasil warna campuran dari kedua warna tersebut. Akhirnya warna yang didapat dari tabel inilah yang akan diletakkan pada screen.

Sekarang kita akan mempelajari proses-proses yang dilakukan oleh T_PUTIMAGE. Kita akan memulai dari vertikal loop saja, karena bagian program di atas vertikal loop sama persis dengan PUTIMAGE yaitu di mana dilakukan perhitungan pembatasan layar.

```
@Loop1:
```

Label untuk vertikal loop.

```
Push Cx
Push di
Push si
```

Simpan counter Vertikal loop, di dan si ke dalam stack.

```
Mov Cx,XRead
```

Isi counter horizontal loop dengan Xread.

```
@Loop2:
```

Label untuk horizontal loop.

```
Lodsb
or al,al
```

Ambil warna pertama dari sprite, test apakah warna tersebut = 0.

```
Jz @No_Write

Push bx
Mov bl,[es:di]
```

Simpan Bx kemudian ambil warna kedua dari screen.

```
Xor bh,bh
Xor ah,ah
Dec al
```

Kurangi index warna pertama. Hal ini dilakukan karena pada saat kita membuat tabel transparant, kita tidak menyertakan warna nomor 0. Tentunya Anda ingat kan? Karena warna 0 tidak akan diproses oleh T_PUTIMAGE ini. Sehingga pada tabel transparant, kolom pertama akan langsung diisi dengan warna nomor 1 bukannya 0.

```
Shl ax,8
Add ax,bx
```

Kalikan Ax dengan 256 kemudian tambahkan dengan Bx. Bx ialah index kedua yang didapat dari warna kedua. Sistem perhitungan ini digunakan untuk memvisualisasikan array 2 dimensi pada assembler. Assembler tidak mengenal array 2 dimensi, melainkan menggunakan array 1 dimensi. Situasi ini seperti halnya alokasi pixel pada layar monitor yang menggunakan sistem offset. Anggap saja layar itu array 2 dimensi berukuran 320 x 200. Untuk melakukan alokasi pixel, assmbler tidak dapat Anda perintah langsung mengisi titik pada koordinat 320 x 200. Anda harus melakukan perhitungan offset yaitu Y*320 + X. Anggap saja Y adalah INDEX PERTAMA dan X adalah INDEX KEDUA maka di sini terjadi hal serupa di mana kita harus mengalikan INDEX PERTAMA dengan 256 kemudian ditambah dengan INDEX KEDUA.

```
Push es
Push di
Les di,PalTable
Add di,ax
Mov al,[es:di]
```

Simpan ES dan DI karena kita akan menggunakannya untuk mengalokasikan tabel PALTABLE.

Kemudian ambil warna yang didapat dengan menggunakan INDEX AX (Index AX ini diperoleh dari hasil perhitungan index gabungan dari INDEX pertama dan INDEX kedua seperti yang telah dijelaskan di atas).

```
Pop di
Pop es
Pop bx
```

Kembalikan DI, ES dan Bx.

```
Stosb
```

Letakkan titik yang didapat dari tabel transparant ke screen.

```
    dec di
  @No_Write:
    inc di
Loop @Loop2
Pop si
Pop di
Add si,Length
Add di,320
Pop Cx
Loop @Loop1
```

Sisa program selanjutnya sama persis dengan procedure PUTIMAGE.

Setelah kita bermain-main dengan transparant, Put satu warna apa salahnya kalau kita mencoba membuat PUT SATU WARNA yang TRANSPARANT. Tidak begitu sulit kan? Kita hanya perlu melakukan sedikit perubahan pada T_PUTIMAGE yang telah kita buat. Selain menggunakan titik pertama yang dibaca dari sprite sebagai INDEX PERTAMA tabel transparant, kita tinggal menggunakan warna yang dikirim lewat parameter sebagai INDEX PERTAMA. Beres kan?

Listing dari T_PUT1COL ini adalah sebagai berikut:

```
Procedure T_Put1Col(Var PicBuff;X1,Y1:Integer;Col:Byte;
Var Buffer;Var PalTable);Assembler;
Var XWStart:Word;
    XRStart:Word;
    YWStart:Word;
    YRStart:Word;
    Length:Word;
    Width:Word;
    XRead:Word;
    YRead:Word;
  Asm
    Push ds
    Lds si,Buffer
    Les di,PicBuff
    Mov Ax,[ds:si]
    Mov Length,Ax
    Mov XRead,Ax
    Add si,2
    Mov Ax,[ds:si]
    Mov Width,Ax
    Mov YRead,Ax
    Mov YRStart,0
    Mov Ax,Y1
    Mov YWStart,Ax
    Add si,2
    Mov XRStart,0
    Mov Ax,X1
    Mov XWStart,Ax
    Cmp Ax,0
    Jge @X1_Not_Negative
      Neg Ax
      Mov XRStart,Ax
      Mov XWStart,0
      Sub XRead,Ax
    @X1_Not_Negative:
      Mov Ax,X1
      Add Ax,Length
      Dec Ax
      Cmp Ax,319
    Jle @X1_No_Right_Crop
      Sub Ax,319
```

```
  Sub XRead,Ax
@X1_No_Right_Crop:
  Mov Ax,Y1
  Cmp Ax,0
Jge @No_Top_Crop
  Neg Ax
  Mov YWStart,0
  Mov YRStart,Ax
  Sub YRead,Ax
@No_Top_Crop:
  Mov Ax,Y1
  Add Ax,Width
  Dec Ax
  Cmp Ax,199
Jle @No_Bot_Crop
  Sub Ax,199
  Sub YRead,Ax
@No_Bot_Crop:
  Add di,XWStart
  Mov Ax,YWStart
  Shl Ax,2
  Add Ax,YWStart
  Shl Ax,6
  Add di,Ax

  Mov Ax,YRStart
  Xor Ah,Ah
  Mul Length
  Add si,Ax
  Add si,XRStart

  Mov Cx,YRead
@Loop1:
  Push Cx
  Push di
  Push si
  Mov Cx,XRead
  @Loop2:
      Lodsb
      or al,al
      Jz @No_Write

      Push bx
      Mov bl,[es:di]
      Xor bh,bh
      Xor ah,ah
      Mov al,col
      Dec al
      Shl ax,8
      Add ax,bx
      Push es
      Push di
      Les di,PalTable
      Add di,ax
```

```
          Mov al,[es:di]
          Pop di
          Pop es
          Pop bx

          Stosb
          dec di
        @No_Write:
          inc di
      Loop @Loop2
      Pop si
      Pop di
      Add si,Length
      Add di,320
      Pop Cx
      Loop @Loop1
    Pop ds
  End;
```

PENJELASAN PROGRAM

Perhatikan bagian program dari T_PUT1COL di bawah ini.

```
@Loop2:
          Lodsb
          or al,al
          Jz @No_Write

          Push bx
          Mov bl,[es:di]
          Xor bh,bh
          Xor ah,ah
          Mov al,col
```

Nah di sini dilakukan pengisian register Al yang berfungsi sebagai INDEX PERTAMA secara langsung dari parameter COL.

```
          Dec al
          Shl ax,8
          Add ax,bx
```

Otomatis kedua INDEX yang digunakan ialah COL-1 dan titik yang akan ditumpuk. Dengan cara ini, program akan mengabaikan warna titik yang ada pada sprite.

Sekarang kita akan mencoba sebuah program yang memanfaatkan teknik tersebut.

```
{-----------------------------------------------}
{ Program: B_TR3.PAS                            }
{ Coder : Agustinus Nalwan                      }
{ Fungsi: Mencoba Put transparant satu warna    }
{-----------------------------------------------}

Uses Crt,Dos,VgaCad;

Var Pic     : Pointer;
    PalTable: Pointer;
    Process: Pointer;
    Back    : Pointer;
    Finish : Boolean;
    X,Y     : Integer;
    Ch      : Char;
    Col1    : Boolean;
Begin
  OpenVga256;
  GetMem(Pic,30000);
  GetMem(PalTable,65280);
  GetMem(Process,64000);
  GetMem(Back,64000);
  Install_Palette('Palette.Pal');
  Install_Transparent(PalTable,'Trans.Tab');
  Load_Picture('Pics1.',Screen^);
  GetImage(Screen^,1,1,108,86,Pic^);
  Load_Picture('Pics2.',Back^);
  X:=50;Y:=50;
  Col1:=False;
  Finish:=False;
  Repeat
    Repeat
      MoveTo(Back^,Process^);
      If Col1 then
        T_PutImage(Process^,X,Y,Pic^,PalTable^) else
          T_Put1Col(Process^,X,Y,12,Pic^,PalTable^);
      MoveTo(Process^,Screen^);
    Until Keypressed;
    Ch:=Readkey;
    Case Ch of
      #80:Inc(Y,3);
      #72:Dec(Y,3);
      #75:Dec(X,3);
      #77:Inc(X,3);
      ' ':Col1:=Not Col1;
      #27:Finish:=True;
    End;
  Until Finish;
  Readln;
End.
```

PENJELASAN PROGRAM

Di sini kita memakai program awan kita yang terdahulu. Sekarang tombol spacebar digunakan untuk men-switch antara T_PUTIMAGE yang menggunakan sistem transparant dan T_PUT1COL yang menggunakan sistem Transparant satu warna. Untuk lebih jelasnya cobalah program tersebut dan cobalah untuk mengganti-ganti sistem peletakan dengan spacebar.

Teknik-teknik di atas sangat penting untuk Anda pahami, karena berikutnya kita akan mempelajari teknik yang *NAJIS* (NAJIS di sini maksudnya rumit lho). Anda sudah siap? Ayo maju terus!!

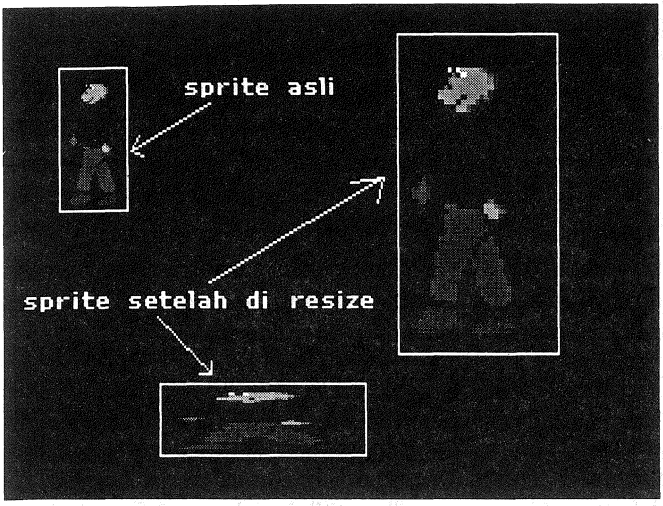
TEKNIK PELETAKAN SPRITE RESIZE

Teknik ini dikenal dengan nama RESIZING. Pada umumnya digunakan oleh pemrogram game 3 Dimensi seperti Wing Commander, After Burner Whacky Wheel, bahkan DOOM pun menggunakan sistem ini (Anda bisa berbangga lho kalau sudah menguasai teknik ini).

Sebenarnya yang dimaksud dengan SPRITE berukuran bebas ini ialah kita dapat meletakkan sprite tersebut dengan panjang dan lebar sesuai keinginan kita. Walaupun sprite tersebut berukuran 50 x 50 tapi kita dapat menampilkannya dalam ukuran 80 x 10. Memang kelihatannya aneh karena sprite tersebut akan terlihat gepeng. Tetapi teknik ini bila Anda gunakan dengan tepat, Anda dapat menerapkannya dalam pembuatan game 3 Dimensi seperti DOOM.

Cara kerja dari teknik ini adalah menggunakan sistem pengulangan atau pengabaian titik dari sprite sehingga ukuran sprite dapat diatur sesuai keinginan kita.

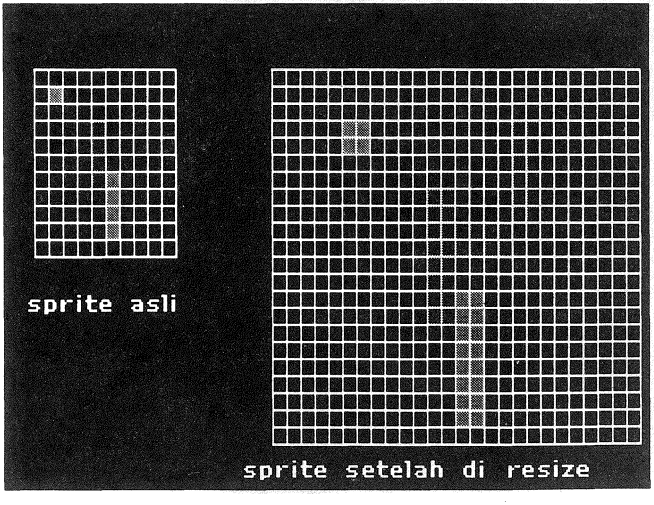
Coba lihat gambar berikut.



Inilah gambaran dari teknik Put dengan menggunakan Resizing. Gambar di sebelah kiri atas itu adalah sprite asli yang disimpan di dalam memori dengan ukuran 30 x 40 titik. Gambar yang di bawah adalah sprite yang diletakkan menggunakan PUT dengan teknik resizing dengan ukuran 80 x 20 titik. Sedangkan gambar yang di sebelah kanan menggunakan ukuran 70 x 100.

Sebenarnya efek pembesaran itu dilakukan melalui pengulangan titik-titik tertentu pada gambar, jika ukuran yang dituju lebih besar dari ukuran sprite. Jika ukuran yang dituju lebih kecil dari ukuran sprite maka dilakukan pelenyapan beberapa titik.

Coba perhatikan gambar berikut ini.



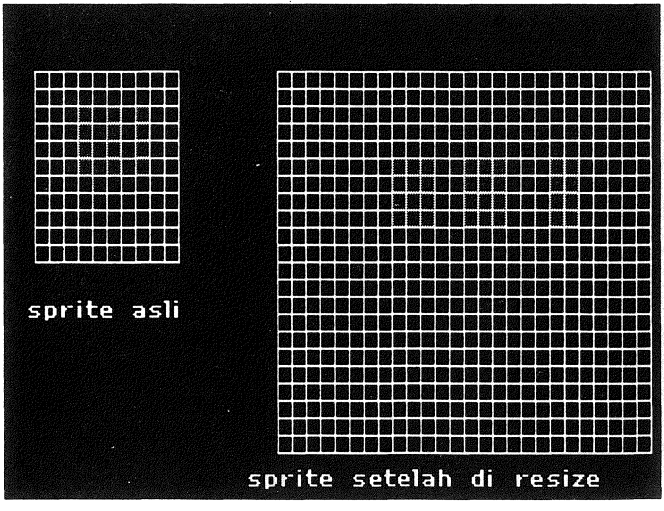
Lihatlah sprite yang asli. Bandingkan dengan sprite setelah dilakukan proses resize. Di sini sprite yang berukuran 8 x 10 diletakkan dengan ukuran 16 x 20. (Jumlah kotak di gambar tersebut tidak menunjukkan ukuran dari sprite !!!). Perhatikan sprite setelah di-resize. Setiap titik pada sprite digambar sebagai sebuah balok berukuran 2 x 2 sehingga gambar menjadi lebih besar 2 kali lipat. Program mengetahui kalau satu titik harus diulang dua kali dari rumus ukuran tujuan dibagi dengan ukuran asli (16/8 = 2).

Perlu diketahui, pembesaran horizontal tidak selalu sama dengan pembesaran vertikal. Keduanya dihitung secara terpisah sehingga bisa terjadi situasi di mana satu titik di-resize menjadi balok berukuran 8 x 2.

Nah sekarang muncul masalah baru, bagaimana jika kita hendak memunculkan sprite tersebut dengan ukuran 12 x 20. Skala sumbu Y tidak mengalami masalah karena 20 /10 = 2. Jadi setiap titik akan dibesarkan secara vertikal menjadi dua titik. Tetapi bagaimana dengan skala sumbu X? 12/8 = 1.5. Kita tidak mungkin menampilkan titik berukuran 1.5 pada layar monitor kan?

Nah untuk ini dilakukan pembesaran horizontal dengan skala yang tidak merata, kadang-kadang titik akan dibesarkan menjadi dua titik, kadang tetap menjadi satu titik. Dengan begini ukuran yang diminta akan terpenuhi. Memang hasil dari pembesaran dengan sistem ini kadang akan menimbulkan efek yang agak jelek pada gambar-gambar berpola tertentu. Tetapi sejauh ini teknik inilah yang digunakan oleh semua game yang membutuhkan teknik sprite resizing.

Untuk lebih jelasnya lihatlah gambar berikut.



Perhatikan sprite setelah di-resize, skala horizontal yang digunakan tidak selalu 3. Kadang-kadang 2 titik atau 3 titik. Dengan cara ini pembesaran yang diinginkan akan tercapai.

Sekarang bagaimana cara kita menentukan kapan sebuah titik diulang tiga kali, kapan diulang dua kali pada situasi di atas? Di sini kita menggunakan sistem pembulatan bilangan pecahan. Misalkan kita hendak me-resize sprite berukuran 4 x 4 menjadi 6 x 8, maka kita cari dahulu skala X dan skala Y-nya.

Skala X dapat dicari dari 4/6 = 0.67. Skala Y dapat dicari dari 4/8 =0.5.

Skala ini nantinya akan digunakan sebagai penambahan koordinat pada koordinat sprite sewaktu dilakukannya proses peletakan titik. Karena ukuran tujuan adalah 6 x 8 maka panjang sprite yang harus dihasilkan ialah 6 titik. Sekarang kita akan menentukan titik pada koordinat sprite berapa yang harus kita masukkan pada keenam titik ini.

Pertama koordinat sprite untuk Y axis yang diambil ialah 0.

- Titik pertama. Koordinat sprite yang diambil ialah 0
- Titik kedua. Koordinat sprite yang diambil ialah 0 + 1*Skala X = 0.67. Setelah dibulatkan maka titik yang diambil ialah titik dengan koordinat sprite 1
- Titik ketiga. Koordinat sprite yang diambil ialah 0 + 2*Skala X = 1.34. Setelah dibulatkan maka titik yang diambil ialah titik dengan koordinat sprite 1
- Titik keempat. Koordinat sprite yang diambil ialah 0 + 3*Skala X = 2.01. Setelah dibulatkan maka titik yang diambil ialah titik dengan koordinat 2
- Titik kelima. Koordinat sprite yang diambil ialah 0 + 4*Skala X = 2.68. Setelah dibulatkan maka titik yang diambil ialah titik dengan koordinat 3
- Titik keenam. Koordinat sprite yang diambil ialah 0 + 5*Skala X = 3.35. Setelah dibulatkan maka titik yang diambil ialah titik dengan koordinat 3.

Proses selanjutnya ialah mengulangi lagi penggambaran titik secara horizontal tetapi dengan koordinat Y berikutnya. Ingat koordinat Y yang diambil juga menggunakan proses yang sama seperti di atas. Berarti koordinat Y berikutnya ialah 0 + 1*Skala Y = 0.5 dibulatkan menjadi 1.

Proses ini diulang terus hingga looping vertikal selesai. Dengan cara ini titik akan terbagi secara merata pada ukuran sprite yang baru. Proses ini pun memiliki satu keuntungan yaitu kita tidak perlu membuat proses tersendiri untuk melakukan pengecilan gambar, karena proses ini berlaku pula untuk penggambaran sprite yang lebih kecil dari ukuran aslinya.

Coba sekarang kita akan menggunakan algoritma di bawah untuk me-resize sprite berukuran 6 x 8 menjadi 4 x 4.

Pertama-tama dilakukan penghitungan Skala X = 6/4 = 1.5 dan
Skala Y = 8/4 = 2.

Koordinat Y sprite pertama yang diambil ialah 0.

- Titik pertama. Koordinat sprite yang diambil ialah 0.
- Titik kedua . Koordinat sprite yang diambil ialah 0 + 1*Skala X = 1.5 dibulatkan menjadi 2.
- Titik ketiga. Koordinat sprite yang diambil ialah 0 + 2*Skala X = 3.
- Titik keempat. Koordinat sprite yang diambil ialah 0 + 3*Skala X = 4.5 dibulatkan menjadi 5.

Proses selanjutnya ialah memproses koordinat Y berikutnya. Koordinat Y berikutnya ialah 0 + 1*Skala Y = 2.

Perhatikan proses di atas, titik nomor 1 dan titik nomor 4 tidak diambil oleh proses tersebut sehingga titik tersebut tidak akan dialokasikan ke screen. Hal ini membuktikan bahwa teori kita di atas dalam melakukan pengecilan gambar, ada beberapa titik yang akan dilenyapkan.

Teknik di atas sangatlah handal dalam menangani masalah resizing gambar, karena dengan satu proses kita dapat melakukan pembesaran ataupun pengecilan gambar.

Sebenarnya jika Anda jeli teknik ini adalah pengembangan dari teknik Bresenham Algorithm yang dahulu kita gunakan untuk menggambar garis miring.

Tentunya kita tidak akan membuat procedure PUT dengan teknik resize ini di Turbo Pascal. Kita akan langsung memprogramnya dalam bahasa Assembler. Ingat, Assembler tidak mengenal bilangan pecahan seperti 1.5 ataupun 0.67. Nah bagaimana pemecahan masalah ini? Inilah kenapa teknik PUT resizing ini saya sebut teknik *NAJIS* karena cara kerjanya cukup rumit dan membingungkan bagi orang yang belum biasa memakai sistem FIXED POINT ALGORITHM. Sebenarnya sistem ini biasa dipakai oleh orang yang

berkecimpung dalam dunia pemrograman 3 Dimensi. Di mana dikenal dua macam sistem yaitu FLOATING POINT ALGORITHM dan FIXED POINT ALGORITHM. Masing-masing memiliki keunggulan dan kelemahan tersendiri. Kita tidak akan membicarakan teknik ini secara mendetil. Tetapi kita akan menggunakan teknik FIXED POINT ALGORITHM ini untuk memecahkan masalah kita, untuk itu akan saya jelaskan sedikit tentang FIXED POINT ALGORITHM ini.

Teknik ini dirancang untuk mengatasi kelemahan ASSEMBLER dalam menangani bilangan pecahan. Segala bilangan yang ada dalam bahasa assembler ialah bilangan bulat. Nah dengan akal licik, kita anggap bilangan 256 itu adalah 1. Maka bilangan 128 adalah 0.5 dan bilangan 64 adalah 0.25. Nah dengan berpatokan bahwa bilangan 256 itu adalah 1 maka kita dapat menghasilkan bilangan pecahan. Misalnya kita hendak mengalikan angka 200 dengan 0.67. Berhubung Assembler tidak mengenal bilangan 0.67, kita anggap saja bilangan 256 = 1 maka 0.67 adalah 172. Kalikan saja 200 dengan 172 = 34400. Ingat hasil perkalian ini ialah bilangan yang menggunakan dasar 256 sebagai 1. Sehingga untuk mengembalikannya ke bilangan desimal yang benar, kita tinggal membagi 34400 dengan 256 = 134.

Hasil perhitungan 200 * 0.67 dengan menggunakan FIXED POINT ALGORITHM ialah 134. Nah sekarang kita coba dengan kalkulator, coba ambil kalkulator dan hitung 200 * 0.67. Hasilnya 134!!

WOW ajaib kan?? He he he he he. Inilah yang dinamakan dengan FIXED POINT ALGORITHM. Semakin besar bilangan yang Anda anggap sebagai 1 semakin tepat hasil perhitungan yang Anda peroleh. Tetapi pada umumnya kita menggunakan 256 sebagai standar. Bilangan yang Anda gunakan sebagai 1 disarankan menggunakan bilangan kelipatan 2 seperti 4, 6, 8, 16 dan sebagainya. Dengan cara ini kita dapat menghindari perkalian dan pembagian, karena perkalian dan pembagian ini membutuhkan waktu yang lama jika dibandingkan dengan perintah Assembler yang lain. Jika Anda menggunakan bilangan kelipatan 2 misalnya 256 maka Anda dapat menggantikan perkalian dengan 256 menjadi SHL 8 (shift kiri 8). Demikian pula pembagian dengan bilangan 256 dapat diganti dengan SHR 8 (shift kanan 8).

Nah sekarang kita kembali ke pokok permasalahan di mana kita menggunakan teknik skala untuk melakukan resize sprite. Skala yang dihasilkan kebanyakan akan berupa bilangan pecahan. Di sinilah teknik FIXED POINT ALGORITHM beraksi. Kita akan mengganti bilangan pecahan tersebut dengan bilangan yang menganggap 256 sebagai 1.

Ok sudah siap?? Inilah procedure PUT yang menggunakan teknik resize.

```
Procedure Put3D(Var PicBuff;X,Y: Integer;XL,YL:Word; Var Buffer);
Var XWStart:Word;
    YWStart:Word;
    Length:Word;
    Width:Word;
    XRead:Word;
    YRead:Word;

    XFixed:Word;
    YFixed:Word;
    XFStart:Word;
    YFStart:Word;
    RX    :Word;
    RY    :Word;
  Begin
  Asm
    Push ds
    Lds si,Buffer
    Les di,PicBuff
    Mov Ax,[ds:si]
    Mov Length,Ax
    Mov Ah,Al
    Xor Al,Al
    Mov Bx,Ax
    Xor Dx,Dx
    Div XL
    Mov Rx,Ax
    Mov Ax,Bx
    Xor Dx,Dx
    Div Rx
    Mov XRead,Ax
    Add si,2
    Mov Ax,[ds:si]
    Mov Width,Ax
    Mov Ah,Al
    Xor Al,Al
    Mov Bx,Ax
    Xor Dx,Dx
    Div YL
    Mov Ry,Ax
    Mov Ax,Bx
    Xor Dx,Dx
    Div Ry
    Mov YRead,Ax
    Mov XFixed,0
    Mov YFixed,0
    Mov XFStart,0
    Mov YFStart,0
    Mov Ax,Y
    Mov YWStart,Ax
    Add si,2
```

```
Mov Ax,X
Mov XWStart,Ax
Cmp Ax,0
Jge @X1_Not_Negative
  Neg Ax
  Mov XWStart,0
  Sub XRead,Ax
  Mul Rx
  Mov XFStart,Ax
@X1_Not_Negative:
  Mov Ax,XWStart
  Add Ax,XRead
  Dec Ax
  Cmp Ax,319
Jle @X1_No_Right_Crop
  Sub Ax,319
  Sub XRead,Ax
@X1_No_Right_Crop:
  Mov Ax,Y
  Cmp Ax,0
Jge @No_Top_Crop
  Neg Ax
  Mov YWStart,0
  Sub YRead,Ax
  Mul Ry
  Mov YFStart,Ax
@No_Top_Crop:
  Mov Ax,YWStart
  Add Ax,YRead
  Cmp Ax,200
Jle @No_Bot_Crop
  Sub Ax,200
  Sub YRead,Ax
@No_Bot_Crop:
  Add di,XWStart
  Mov Ax,YWStart
  Shl Ax,2
  Add Ax,YWStart
  Shl Ax,6
  Add di,Ax

  Mov Cx,YRead
  Mov Ax,YFStart
  Mov YFixed,Ax
@Loop1:
  Push Cx
  Push di
  Push si
  Mov Ax,YFixed
  Mov Al,Ah
  Xor Ah,Ah
  Mul Length
  Add si,Ax
```

```
      Mov Cx,XRead
      Mov Ax,XFStart
      Mov XFixed,Ax
      Mov Bx,Rx
      @Loop2:
         Mov Ax,XFixed
         Mov Al,Ah
         Xor Ah,Ah
         Push si
         Add si,Ax

         Lodsb
         Pop si
         Add XFixed,Bx
         or al,al
         Jz @No_Write
         Stosb
         dec di
       @No_Write:
         inc di
      Loop @Loop2
      Pop si
      Pop di
      Mov Bx,Ry
      Add YFixed,Bx
      Add di,320
      Pop Cx
      Loop @Loop1
    Pop ds
  End;
  End;
```

PENJELASAN PROGRAM

Lumayan panjang. Tapi tidak terlalu sulit kok, jangan khawatir. Yuk sekarang kita bahas procedure di atas satu persatu.

```
Procedure Put3D(Var PicBuff;X,Y: Integer;XL,YL:Word; Var Buffer);
Var XWStart:Word;
    YWStart:Word;
    Length:Word;
    Width:Word;
    XRead:Word;
    YRead:Word;

    XFixed:Word;
    YFixed:Word;
    XFStart:Word;
    YFStart:Word;
    RX    :Word;
```

```
    RY    :Word;
  Begin
  Asm
    Push ds
    Lds si,Buffer
    Les di,PicBuff

Arahkan DS:SI ke sprite.
Arahkan ES:DI ke screen.

    Mov Ax,[ds:si]
    Mov Length,Ax
```

Ambil panjang sprite dan masukkan ke variabel Length.

```
    Mov Ah,Al
    Xor Al,Al
```

Proses di atas ini sama halnya dengan Ax * 256 = Ax atau sama juga dengan SHL Ax,8. Karena dengan memindahkan low register ke high register yang 8 bit lebih tinggi sama juga dengan menggeser low register ke high register. Hal ini dilakukan untuk optimasi kecepatan. Karena perintah MOV lebih cepat daripada SHL ataupun perkalian.

Di sini dilakukan perkalian terhadap panjang dari sprite yang bertujuan untuk menciptakan sistem bilangan FIXED POINT yang menganggap 256 sebagai 1 untuk itu kita perlu mengalikan panjang sprite dengan 256.

```
    Mov Bx,Ax
    Xor Dx,Dx
    Div XL
```

Kemudian bagi panjang sprite (dalam satuan 256) dengan panjang tujuan yaitu XL yang dikirim lewat parameter.

```
    Mov Rx,Ax
```

Masukkan hasilnya ke Rx. Rx inilah yang berfungsi sebagai Skala X.

```
    Mov Ax,Bx
    Xor Dx,Dx
    Div Rx
    Mov XRead,Ax
```

Nah sekarang kita harus menentukan berapa kali kita harus melakukan peletakan titik secara horizontal.

Caranya ialah dengan membagi panjang sprite dengan Rx. Hasilnya dimasukkan ke dalam variabel Xread.

```
Add si,2
Mov Ax,[ds:si]
Mov Width,Ax
Mov Ah,Al
Xor Al,Al
Mov Bx,Ax
Xor Dx,Dx
Div YL
```

Ambil lebar dari sprite. Kemudian dilakukan proses yang sama seperti proses perhitungan skala X. Dengan membagi lebar sprite dengan lebar tujuan, kita akan memperoleh skala Y.

```
Mov Ry,Ax
Mov Ax,Bx
Xor Dx,Dx
Div Ry
Mov YRead,Ax
```

Yread diperoleh dengan cara yang sama seperti kita mencari Xread. Yread ini menunjukkan jumlah peletakan titik secara vertikal.

```
Mov XFixed,0
Mov YFixed,0
Mov XFStart,0
Mov YFStart,0
```

Nah sekarang kita masuk ke bagian yang paling *NAJIS*. Di sini kita mesti melakukan pembatasan layar terhadap sprite yang telah di-resize ini. Pada bagian ini konsentrasilah sungguh-sungguh karena ini adalah bagian yang tersulit dari procedure ini.

Variabel XFStart, YFStart di-nol-kan terlebih dahulu dengan asumsi tidak ada bagian dari sprite yang melebihi batas layar.

```
Mov Ax,Y
Mov YWStart,Ax
```

Masukkan koordinat Y ke dalam YWStart. Koordinat ini menunjukkan koordinat Y dari sisi atas sprite setelah di-resize.

```
Add si,2
Mov Ax,X
Mov XWStart,Ax
```

Masukkan koordinat X ke dalam XWStart.

```
Cmp Ax,0
```

Deteksi apakah sprite melebihi batas kiri layar.

```
Jge @X1_Not_Negative
  Neg Ax
  Mov XWStart,0
  Sub XRead,Ax
  Mul Rx
  Mov XFStart,Ax
```

Jika sprite melebihi batas kiri layar, positifkan Ax. Kemudian sprite yang telah di-resize harus dipotong sebanyak Ax titik, lalu Xread dikurangi dengan Ax. XWStart yang menunjukkan koordinat sisi kiri sprite diset menjadi 0 karena terjadi proses pembatasan kiri pada sprite.

Jumlah titik yang terpotong oleh batas kiri layar ini selanjutnya dikalikan oleh Rx (skala X) sehingga kita akan memperoleh koordinat sprite horizontal awal dari PEMBACAAN TITIK pada sprite. Data ini disimpan dalam XFStart berupa bilangan FIXED POINT (256 = 1).

```
@X1_Not_Negative:
  Mov Ax,XWStart
  Add Ax,XRead
  Dec Ax
  Cmp Ax,319
```

Deteksi batas kanan layar, caranya ialah dengan menambahkan XWStart yang menunjukkan start horizontal pertama peletakan titik dengan Xread yang menunjukkan jumlah titik yang akan diletakkan.

Jika hasilnya melebihi 319 maka sprite menerobos batas kanan layar.

```
Jle @X1_No_Right_Crop
  Sub Ax,319
  Sub XRead,Ax
```

Sprite menerobos batas kanan layar. Kurangi jumlah peletakan titik horizontal dengan jumlah titik yang menerobos batas kanan layar.

```
@X1_No_Right_Crop:
  Mov Ax,Y
  Cmp Ax,0
```

Deteksi apakah sprite menerobos atas layar.

```
Jge @No_Top_Crop
  Neg Ax
  Mov YWStart,0
  Sub YRead,Ax
  Mul Ry
  Mov YFStart,Ax
```

Sprite menerobos atas layar maka positifkan koordinat Ax, kemudian set YWStart yang merupakan start pertama peletakan titik secara vertikal menjadi 0 karena terjadi pembatasan atas layar. Selanjutnya kurangi jumlah peletakan titik secara vertikal yaitu Yread dengan jumlah titik yang menerobos batas atas layar yaitu Ax. Kemudian dilakukan proses penghitungan koordinat sprite awal dari pembacaan titik pada sprite secara vertikal yaitu dengan mengalikan jumlah titik yang menerobos batas atas layar dengan Ry (Skala Y). Hasilnya disimpan dalam variabel YFStart berupa FIXED POINT (256 = 1).

```
@No_Top_Crop:
  Mov Ax,YWStart
  Add Ax,YRead
  Cmp Ax,200
```

Deteksi apakah sprite menerobos batas bawah layar.

```
Jle @No_Bot_Crop
  Sub Ax,200
  Sub YRead,Ax
```

Sprite menerobos batas bawah layar. Kurangi Yread dengan jumlah titik yang menerobos batas bawah layar.

```
@No_Bot_Crop:
  Add di,XWStart
  Mov Ax,YWStart
  Shl Ax,2
  Add Ax,YWStart
  Shl Ax,6
  Add di,Ax
```

Lakukan proses perhitungan offset terhadap koordinat (XWStart, YWStart) kemudian tambahkan ke DI yang merupakan offset dari screen.

```
  Mov Cx,YRead
  Mov Ax,YFStart
  Mov YFixed,Ax
```

Isikan Yread sebagai counter vertikal loop ke dalam CX.

Isi variabel Yfixed dengan YFStart. Variabel Yfixed ini berisi koordinat sprite Y dalam bentuk bilangan FIXED POINT (256 = 1).

```
@Loop1:
```

Label untuk vertikal loop.

```
  Push Cx
```

```
Push di
Push si
Mov Ax,YFixed
Mov Al,Ah
Xor Ah,Ah
Mul Length
Add si,Ax
```

Simpan Cx, DS, SI ke dalam stack. Kemudian hitung jumlah titik yang dilompati pada saat melakukan pembacaan pada sprite. Caranya ialah dengan mengubah dahulu bilangan Yfixed yang merupakan bilangan FIXED POINT (256 = 1) menjadi bilangan biasa dengan membaginya dengan 256 atau dengan memindahkan high register ke low register. Kemudian kalikan dengan panjang gambar. Hasil perhitungan ini menunjukkan jumlah titik yang akan dilompati saat melakukan pembacaan titik pada sprite. Oleh sebab itu hasil tersebut ditambahkan ke register SI sebagai offset dari sprite.

```
Mov Cx,XRead
Mov Ax,XFStart
Mov XFixed,Ax
Mov Bx,Rx
```

Isikan Counter horizontal loop dari Xread.

Isikan pula ke dalam Xfixed, start awal XFStart yang menunjukkan titik horizontal pertama yang akan ditampilkan ke screen setelah dilakukannya proses pembatasan layar. Isikan harga Rx ke dalam register Bx untuk mempercepat proses di dalam loop nantinya karena pengaksesan register jauh lebih cepat daripada pengaksesan variabel luar.

```
@Loop2:
```

Label untuk horizontal loop.

```
Mov Ax,XFixed
Mov Al,Ah
Xor Ah,Ah
Push si
Add si,Ax
```

Ubah sistem bilangan FIXED POINT pada Xfixed menjadi bilangan biasa untuk mengetahui harga asli dari bilangan tersebut. Kemudian tambahkan SI dengan bilangan tersebut. Teknik ini sama persis dengan yang telah dijelaskan pada bagian terdahulu di mana kita menggunakan 0 + 1*Skala X.

```
Lodsb
```

Ambil titik pada koordinat sprite tersebut.

```
Pop si
Add XFixed,Bx
```

Tambahkan Xfixed dengan skala X. Ini digunakan untuk menggantikan fungsi perkalian dengan skala X pada beberapa halaman terdahulu yang kita lakukan. Anda masih ingat kan sewaktu kita memakai 1* Skala X , 2*Skala X. 3* Skala X dan seterusnya. Di sini kita menghindari proses perkalian. Untuk itu kita menggunakan variabel Xfixed sebagai tempat penampung dari penjumlahan terhadap Skala X pada setiap horizontal loop.

```
or al,al
Jz @No_Write
Stosb
dec di
```

Test apakah titik itu berwarna 0. Kalau tidak maka letakkan titik tersebut.

```
@No_Write:
  inc di
Loop @Loop2
```

Loop ke horizontal loop.

```
Pop si
Pop di
Mov Bx,Ry
Add YFixed,Bx
Add di,320
```

Kembalikan SI dan DI. Kemudian tambahkan Yfixed dengan skala Y untuk meng-akses koordinat sprite Y berikutnya. Tambahkan di dengan 320 untuk melakukan peletakan titik pada koordinat layar Y berikutnya.

```
Pop Cx
```

Kembalikan counter vertikal loop.

```
Loop @Loop1
```

Loop ke vertikal loop.

```
Pop ds
```

Kembalikan DS.

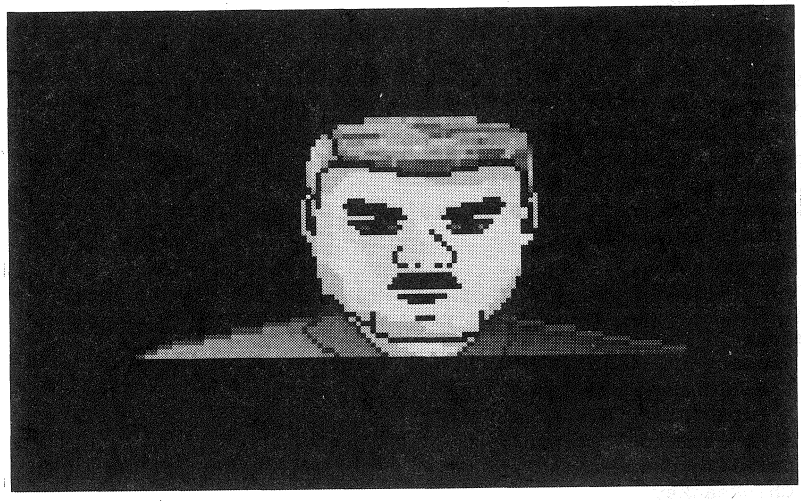
```
End;
End;
```

Memang teknik ini cukup rumit. Tapi dengan adanya procedure ini kita dapat menciptakan efek-efek sprite yang sangat bagus yang biasa dipakai pada game-game modern saat ini. Sekarang marilah kita membuat sebuah program yang memanfaatkan fasilitas ini. Sekarang kita tidak memakai awan di atas hutan lagi kok (he he he). Saya akan buatkan gambar yang lain supaya pembaca tidak bosan.

```
{------------------------------------}
{ Program: B_3D1.PAS                 }
{ Coder : Agustinus Nalwan           }
{ Fungsi: Mencoba Put Resize         }
{------------------------------------}

Uses Crt,Dos,VgaCad;

Var Pic:Pointer;
    Luar:Boolean;
    Ax,Ay:Word;
    Proc:Pointer;
    X,Y:Integer;
    Ch:Char;

Begin
  OpenVga256;
  Install_Palette('Palette.pal');
  Load_Picture('Pics3.',Screen^);
  GetMem(Pic,30000);
  GetImage(Screen^,1,1,110,45,Pic^);
  GetMem(Proc,64000);
  Luar:=False;
  X:=100;Y:=100;
  AX:=110;AY:=45;
  Repeat
    Repeat
      FillChar(Proc^,64000,1);
      Put3D(Proc^,X,Y,AX,AY,Pic^);
      MoveTo(Proc^,Screen^);
    Until Keypressed;
    Ch:=Readkey;
    Case Ch of
      #80:Inc(Y,5);
      #75:Dec(X,5);
      #77:Inc(X,5);
      #72:Dec(Y,5);
      'd':Inc(AX,5);
      'a':Dec(AX,5);
      'w':Dec(AY,5);
      'x':Inc(AY,5);
      #27:Luar:=True;
    End;
  Until Luar;
  Readln;
End.
```

PENJELASAN PROGRAM

Struktur program di atas masih memakai struktur program kita yang terdahulu. Apabila Anda jalankan, program ini akan menampilkan gambar kepala orang. Untuk menggerakkan orang tersebut Anda dapat menggunakan kursor dan untuk mengubah ukuran sprite Anda dapat menggunakan tombol A,X,D,W huruf kecil. Tombol D dan X untuk memperbesar sprite secara horizontal dan vertikal. Tombol A dan W untuk memperkecil sprite secara horizontal dan vertikal. Ingat jangan mengecilkan sprite hingga lebih kecil dari satu garis, karena procedure PUT3D akan tetap memproses sprite itu menjadi lebih kecil dari satu garis dan akan mengalami error. Cobalah bermain-main sebentar dengan PUT3D ini. Cobalah untuk menggeser sprite hingga melebihi batas layar. Bagaimana? Tidaklah sia-sia usaha Anda untuk mempelajari teknik di atas yang boleh dikata sangat rumit.



Gambar 1.3 Output dari B_3D1.PAS

Tentunya sekarang Anda ingin teknik yang lebih menantang lagi bukan? Bagaimana dengan PUT yang dapat di-resize dan juga TRANSPARANT!! He he he makin lama makin rumit nih!!

Sebenarnya procedure-procedure pada unit VGACAD.TPU sudah dirancang sedemikian, sehingga bila ada perubahan tidak perlu mengubah logika. Kita cukup mengganti bagian-bagian tertentu saja. Nah untuk melakukan resizing dan transparant, kita tinggal menyisipkan sepotong algoritma transpa-

rant untuk menggantikan peletakan titik pada loop yang terdalam. Put transparant ini akan kita beri nama T_PUT3D.

```
Procedure T_Put3D(Var PicBuff;X,Y: Integer;XL,YL:Word;
Var Buffer;Var PalTable);
Var XWStart:Word;
    YWStart:Word;
    Length:Word;
    Width:Word;
    XRead:Word;  { CX }
    YRead:Word;

    XFixed:Word;
    YFixed:Word;
    XFStart:Word;
    YFStart:Word;
    RX    :Word;
    RY    :Word;
  Begin
  Asm
    Push ds
    Lds si,Buffer
    Les di,PicBuff
    Mov Ax,[ds:si]
    Mov Length,Ax
    Mov Ah,Al
    Xor Al,Al
    Mov Bx,Ax
    Xor Dx,Dx
    Div XL
    Mov Rx,Ax
    Mov Ax,Bx
    Xor Dx,Dx
    Div Rx
    Mov XRead,Ax
    Add si,2
    Mov Ax,[ds:si]
    Mov Width,Ax
    Mov Ah,Al
    Xor Al,Al
    Mov Bx,Ax
    Xor Dx,Dx
    Div YL
    Mov Ry,Ax
    Mov Ax,Bx
    Xor Dx,Dx
    Div Ry
    Mov YRead,Ax
    Mov XFixed,0
    Mov YFixed,0
    Mov XFStart,0
    Mov YFStart,0
    Mov Ax,Y
    Mov YWStart,Ax
```

```
Add si,2
Mov Ax,X
Mov XWStart,Ax
Cmp Ax,0
Jge @X1_Not_Negative
  Neg Ax
  Mov XWStart,0
  Sub XRead,Ax
  Mul Rx
  Mov XFStart,Ax
@X1_Not_Negative:
  Mov Ax,XWStart
  Add Ax,XRead
  Dec Ax
  Cmp Ax,319
Jle @X1_No_Right_Crop
  Sub Ax,319
  Sub XRead,Ax
@X1_No_Right_Crop:
  Mov Ax,Y
  Cmp Ax,0
Jge @No_Top_Crop
  Neg Ax
  Mov YWStart,0
  Sub YRead,Ax
  Mul Ry
  Mov YFStart,Ax
@No_Top_Crop:
  Mov Ax,YWStart
  Add Ax,YRead
  Cmp Ax,200
Jle @No_Bot_Crop
  Sub Ax,200
  Sub YRead,Ax
@No_Bot_Crop:
  Add di,XWStart
  Mov Ax,YWStart
  Shl Ax,2
  Add Ax,YWStart
  Shl Ax,6
  Add di,Ax

  Mov Cx,YRead
  Mov Ax,YFStart
  Mov YFixed,Ax
@Loop1:
  Push Cx
  Push di
  Push si
  Mov Ax,YFixed
  Mov Al,Ah
  Xor Ah,Ah
  Mul Length
  Add si,Ax
```

```
    Mov Cx,XRead
    Mov Ax,XFStart
    Mov XFixed,Ax
    Mov Bx,Rx
    @Loop2:
       Mov Ax,XFixed
       Mov Al,Ah
       Xor Ah,Ah
       Push si
       Add si,Ax

       Lodsb
       Pop si
       Add XFixed,Bx
       or al,al
       Jz @No_Write
        { Transparent Algorithm }
        Push bx
        Mov bl,[es:di]
        Xor bh,bh
        Xor ah,ah
        Dec al
        Shl ax,8
        Add ax,bx
        Push es
        Push di
        Les di,PalTable
        Add di,ax
        Mov al,[es:di]
        Pop di
        Pop es
        Pop bx
        { ----------}
       Stosb
       dec di
     @No_Write:
       inc di
    Loop @Loop2
    Pop si
    Pop di
    Mov Bx,Ry
    Add YFixed,Bx
    Add di,320
    Pop Cx
    Loop @Loop1
  Pop ds
End;
End;
```

PENJELASAN PROGRAM

Benar kan? Algoritmanya tidak banyak berbeda. Letak perbedaannya ialah pada saat meletakkan titik pada layar, diganti dengan melakukan proses hashing pada tabel transparant kemudian meletakkan titik yang ditemukan pada tabel ke screen.

Lihatlah bagian algoritma dari transparant berikut.

```
@Loop2:
    Mov Ax,XFixed
    Mov Al,Ah
    Xor Ah,Ah
    Push si
    Add si,Ax

    Lodsb
    Pop si
    Add XFixed,Bx
    or al,al
    Jz @No_Write
     { Transparent Algorithm }
     Push bx
     Mov bl,[es:di]
```

Ambil titik pada screen sebagai INDEX KEDUA.

```
     Xor bh,bh
     Xor ah,ah
     Dec al
     Shl ax,8
     Add ax,bx
```

Di sini dilakukan proses indexing dari tabel transparant seperti cara yang telah kita pelajari pada T_PUTIMAGE terdahulu.

```
     Push es
     Push di
     Les di,PalTable
     Add di,ax
     Mov al,[es:di]
```

Ambil titik dari tabel transparant.

```
     Pop di
     Pop es
     Pop bx
     { ----------}
    Stosb
    dec di
@No_Write:
```

```
        inc di
      Loop @Loop2
```

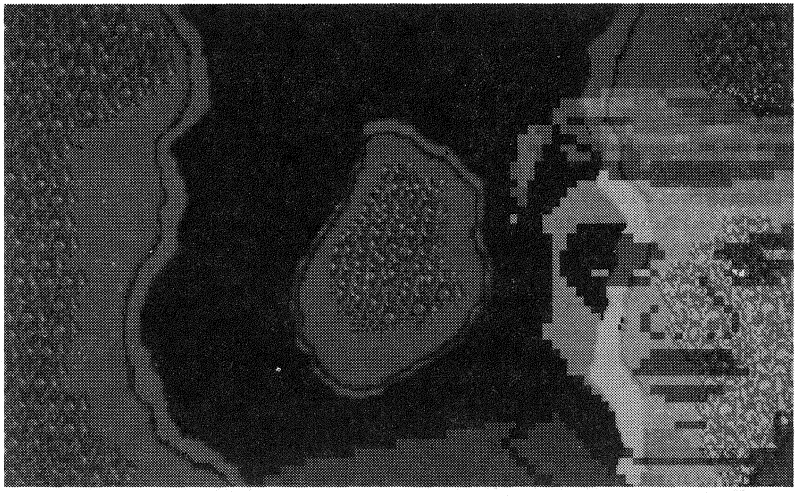
Sekarang kita akan mencoba sebuah program yang akan memanfaatkan teknik di atas.

```
{-------------------------------------------------------------}
{ Program: B_3D2.PAS                                          }
{ Coder  : Agustinus Nalwan                                   }
{ Fungsi: Mencoba Put Resize dengan transparant  }
{-------------------------------------------------------------}

Uses Crt,Dos,VgaCad;

Var Pic:Pointer;
    Luar:Boolean;
    Ax,Ay:Word;
    Proc:Pointer;
    X,Y:Integer;
    Ch:Char;
    PalTable:Pointer;
    Back:Pointer;

Begin
  OpenVga256;
  Install_Palette('Palette.pal');
  Load_Picture('Pics3.',Screen^);
  GetMem(Pic,30000);
  GetImage(Screen^,1,1,110,45,Pic^);
  GetMem(Proc,64000);
  GetMem(PalTable,65280);
  Install_Transparent(PalTable,'Trans.Tab');
  GetMem(Back,64000);
  Load_Picture('Pics2.',Back^);
  Luar:=False;
  X:=100;Y:=100;
  AX:=110;AY:=45;
  Repeat
    Repeat
      MoveTo(Back^,Proc^);
      T_Put3D(Proc^,X,Y,AX,AY,Pic^,PalTable^);
      MoveTo(Proc^,Screen^);
    Until Keypressed;
    Ch:=Readkey;
    Case Ch of
      #80:Inc(Y,5);
      #75:Dec(X,5);
      #77:Inc(X,5);
      #72:Dec(Y,5);
      'd':Inc(AX,5);
      'a':Dec(AX,5);
      'w':Dec(AY,5);
      'x':Inc(AY,5);
      #27:Luar:=True;
```

```
    End;
  Until Luar;
  Readln;
End.
```

PENJELASAN PROGRAM

Coba jalankan program di atas. Anda akan mendapatkan gambar kepala orang tersebut diletakkan secara transparant di atas gambar hutan. Cobalah gerakkan sprite tersebut kemudian ubahlah ukurannya. WOW!! Inilah yang Anda lihat pada permainan HERETIC buatan Raven Software yang merupakan salah satu DOOM Clone (permainan bertipe DOOM). Kita secara perlahan akan mempelajari teknik-teknik yang lebih canggih lagi pada bab-bab berikutnya. Untuk itu pastikan Anda telah menguasai apa yang telah dibahas pada bab ini.



Gambar 1.3b Output dari B_3D2.PAS

BAB 2

PEMBENAHAN KEDUA VGACAD.TPU

Setelah kita melakukan pembenahan pada fasilitas peletakan gambar, pada bab ini kita akan membahas mengenai kompresi data yang juga tak kalah pentingnya dalam pembuatan sebuah game. Kompresi data ini ialah suatu teknik untuk meminimalkan memori yang diperlukan dengan cara menyimpan data dalam bentuk yang sedemikian rupa.

Teknik kompresi data ini banyak macamnya. Yang dapat Anda lihat secara nyata ialah program-program seperti PKZIP.EXE atau ARJ.EXE. Dengan program ini Anda dapat mengkompresi file menjadi file berukuran lebih kecil, bahkan kadang-kadang menjadi jauh lebih kecil dari ukuran aslinya. Teknik ini dapat kita terapkan untuk penyimpanan sprite pada memori sehingga kebutuhan memori pada game menjadi lebih sedikit. Dengan demikian kita dapat memanfaatkan sisa memori yang ada untuk keperluan lain.

VGACAD.TPU yang baru ini akan kita tambah dengan fasilitas-fasilitas kompresi data yang menggunakan berbagai macam teknik. Yang perlu Anda perhatikan ialah routine untuk un-kompres harus dibuat secepat mungkin, karena routine ini akan dipergunakan terus untuk membuka data sprite yang dikompres pada setiap kali program akan melakukan peletakan sprite pada layar monitor. Jika program un-kompres ini lambat, kecepatan game akan menjadi lambat pula. Untuk itu kita harus menciptakan teknik kompresi yang tidak terlalu repot dalam melakukan proses un-kompres. Sebaliknya program untuk mengkompres sprite tidak perlu terlalu cepat, karena program ini hanya digunakan untuk mengkompres sprite kemudian untuk selanjutnya program ini tidak akan diikutsertakan pada looping permainan.

TEKNIK COLOR COMPRESSION

Teknik ini dikenal dengan nama color compress. Sistem kompresinya ialah sebagai berikut.

Andaikan kita memiliki sprite berukuran 15 x 20 dan data byte dari sprite tersebut bila kita lihat adalah:

15 0 20 0 1 1 1 1 2 4 6 6 6 6 8 2 1 1 1 1 9 9 9 9 9 9 9 9 9 9

Dengan sistem kompresi seluruh warna, data tersebut akan diubah menjadi:

15 1 0 1 20 1 0 1 1 4 2 1 4 1 6 4 8 1 2 1 1 4 9 10

Maksud dari hasil kompresi tersebut adalah:

- angka 15 diulang 1 kali
- angka 0 diulang 1 kali
- angka 20 diulang 1 kali
- demikian seterusnya.

Jadi byte pertama menunjukkan angka, sedangkan byte kedua menunjukkan jumlah dari angka tersebut. Demikian seterusnya data disusun berpasang-pasangan. Ingat, jika ada angka yang diulang sebanyak lebih dari 255 kali, angka tersebut harus diuraikan menjadi dua pasangan karena 1 byte maksimum hanya dapat menampung angka 0 hingga 255.

Kejadian akan terjadi bila kita akan mengkompresi angka 5 yang diulang 300 kali. Kita tidak dapat menyimpannya dalam bentuk 5 300 karena 300 telah melebihi 1 byte. Kita harus menyimpannya dalam dua pasangan data sebagai berikut 5 255 5 45.

Untuk lebih jelasnya marilah kita melihat program kompresi tersebut.

```
Procedure Color_Compress(Var Source;Var Destination;Var Size:Word);
Var SegS,OffS :Word;
    SegD,OffD :Word;
    Length,Width:Word;
    Total     :Word;
    Finish    :Boolean;
    Value     :Byte;
    Long      :Byte;
  Begin
    GetAddr(Source,SegS,OffS);
    GetAddr(Destination,SegD,OffD);
    Move(Mem[SegS:OffS],Length,2);
    Move(Mem[SegS:OffS+2],Width,2);
    Total:=Width*Length+4;
    Finish:=False;

    Value:=Mem[SegS:OffS];
    Long:=0;
    Inc(OffD,2);
    Size:=0;
    Repeat
      If (Mem[SegS:OffS]=Value) and (Long<255) and (OffS<Total) then
        Begin
          Inc(Long);
          Inc(OffS);
        End Else
        Begin
          Mem[SegD:OffD]:=Value;
          Inc(OffD);
          Mem[SegD:OffD]:=Long;
          Inc(OffD);
```

```
        Inc(Size,2);
        Value:=Mem[SegS:OffS];
        Long:=0;
        If OffS>=Total then Finish:=True;
      End;
  Until Finish;
  Move(Size,Destination,2);
  Inc(Size,2);
End;
```

PENJELASAN PROGRAM

Seperti yang telah disebutkan di atas karena program kompresi ini tidak perlu terlalu cepat, maka kita menghindari pemakaian routine Assembler (semakin jauh dari Assembler semakin baik supaya tidak terlalu pusing). Procedure ini bertugas untuk mengkompresi data yang dikirimkan lewat parameter Source ke alamat memori yang dikirimkan oleh parameter Destination. Jumlah data hasil kompresi ini akan dikirimkan lewat parameter Size.

```
Procedure Color_Compress(Var Source;Var Destination;Var Size:Word);
Var SegS,OffS :Word;
    SegD,OffD :Word;
    Length,Width:Word;
    Total     :Word;
    Finish    :Boolean;
    Value     :Byte;
    Long      :Byte;
  Begin
    GetAddr(Source,SegS,OffS);
    GetAddr(Destination,SegD,OffD);
```

Fungsi GetAddr ini merupakan fungsi tambahan dari VGACAD.TPU yang bertujuan untuk memecah suatu alamat pointer menjadi Segment dan Offset yang akan dikirimkan kembali melalui parameter. Karena pembuatannya sangat mudah, fungsi GetAddr ini tidak akan saya jelaskan di sini.

Pertama-tama kita harus mengambil segment dan offset dari Source dan Destination.

```
    Move(Mem[SegS:OffS],Length,2);
    Move(Mem[SegS:OffS+2],Width,2);
    Total:=Width*Length+4;
```

Hitung jumlah data yang hendak dikompres. Cara perhitungannya sama persis dengan perhitungan menggunakan fungsi IMAGESIZE pada VGACAD.TPU.

```
Finish:=False;
Value:=Mem[SegS:OffS];
```

Pertama-tama kita ambil angka pertama dari data.

```
Long:=0;
Inc(OffD,2);
Size:=0;
```

Long yang menunjukkan jumlah angka tersebut kita set menjadi 0 dulu dengan asumsi belum ada data berikutnya yang sama dengan value sekarang.

Size kita set menjadi 0 terlebih dahulu.

INC(OFFD,2) berguna untuk mengosongkan tempat 2 byte pada awal Destination guna menampung ukuran dari sprite setelah dikompresi.

```
Repeat
  If (Mem[SegS:OffS]=Value) and (Long<255) and (OffS<Total) then
    Begin
      Inc(Long);
      Inc(OffS);
```

Jika ternyata data berikutnya sama dengan value dan jumlah data belum melebihi 255 serta masih ada data yang perlu diakses, maka tambahkan jumlah data yaitu pada variabel long. Kemudian tambahkan offset source untuk memeriksa data berikutnya. (Situasi ini menunjukkan adanya perulangan data.)

```
    End Else
    Begin
      Mem[SegD:OffD]:=Value;
      Inc(OffD);
      Mem[SegD:OffD]:=Long;
      Inc(OffD);
      Inc(Size,2);
      Value:=Mem[SegS:OffS];
      Long:=0;
      If OffS>=Total then Finish:=True;
```

Jika perulangan data telah selesai maka masukkan value dan jumlah value ke dalam memori tujuan. Jangan lupa tambahkan size dengan 2 karena setiap kali ada penambahan data baru berarti penambahan sebuah pasangan data yang besarnya 2 byte (value dan panjangnya). Set kembali long ke 0 untuk memulai pemeriksaan data baru dan ambil harga Value yang baru dari data berikutnya.

Jika Offset telah mencapai total data maka keluar dari proses kompresi/proses kompresi telah selesai.

```
      End;
    Until Finish;
```

Loop hingga selesai.

```
    Move(Size,Destination,2);
    Inc(Size,2);
```

Masukkan size yang merupakan besar data setelah dikompres pada awal memori Destination sebagai pemberi informasi pada saat kita akan melakukan proses un-kompres.

```
  End;
```

Nah sekarang kita akan mempelajari bagaimana cara membuka data yang dikompresi menggunakan sistem Color Compress tersebut. Program untuk membukanya sengaja dibuat dengan bahasa Assembler untuk mempercepat proses animasi. Adapun cara kerjanya ialah tinggal mengulang setiap data pada nomor genap sejumlah angka yang berada pada 1 nomor berikutnya karena data tersebut disusun dalam pasangan-pasangan.

Listing programnya adalah sebagai berikut.

```
Procedure Color_UnCompress(Var Source;Var Destination);Assembler;
  Asm
    Push ds
    Lds si,Source
    Les di,Destination
    LodsW
    Shr Ax,1
    Mov Cx,Ax
    @Loop1:
      Push Cx
      LodsW
      Xor Ch,Ch
      Mov Cl,Ah
      @Loop2:
        Stosb
      Loop @Loop2
      Pop Cx
    Loop @Loop1
    Pop ds
  End;
```

PENJELASAN PROGRAM

Procedure ini memiliki dua buah parameter. Parameter pertama ialah Source yang berisi alamat memori dari data yang telah dikompres dan Destination yang berisi alamat memori tempat data akan di-un-kompres.

```
Procedure Color_UnCompress(Var Source;Var Destination);Assembler;
```

```
Asm
  Push ds
  Lds si,Source
  Les di,Destination
```

Simpan DS ke dalam stack
Arahkan DS:SI ke Source.
Arahkan ES:DO ke Destination.

```
  LodsW
```

Baca size dari data yang akan di-un-kompres.

```
  Shr Ax,1
```

Bagi dengan 2 karena kita nantinya akan membaca pasangan data tersebut 2 byte demi 2 byte.

```
  Mov Cx,Ax
```

Masukkan ke counter CX.

```
  @Loop1:
    Push Cx
```

Simpan CX ke stack.

```
    LodsW
```

Ambil pasangan data baru = 1 Word = 2 Bytes.

```
    Xor Ch,Ch
    Mov Cl,Ah
```

Isikan counter CX dengan Ah yang berisi jumlah dari value.
Al sendiri menyimpan harga value.

```
    @Loop2:
      Stosb
```

Secara terus-menerus mengulangi proses pengisian Destination dengan Value sejumlah CX kali.

```
    Loop @Loop2
    Pop Cx
```

Kembalikan counter CX untuk mengakses counter pasangan data.

```
  Loop @Loop1
```

Loop hingga seluruh data telah di-un-kompres.

```
    Pop ds
  End;
```

Nah tidak terlalu rumit bukan? Kita harus mengusahakan teknik tercepat dalam melakukan proses un-kompres demi memperoleh kecepatan semaksimal mungkin dalam pembuatan game kita.

Inilah program kompresi kita yang pertama. Sekarang kita akan mencoba membuat sebuah program yang akan memanfaatkan fasilitas ini.

```
{-------------------------------------------------}
{ Program: B_COMP1.PAS                            }
{ Coder : Agustinus Nalwan                        }
{ Fungsi: Mengkompres data gambar dengan metode   }
{         Color Compress                          }
{-------------------------------------------------}

Uses Crt,Dos,VgaCad;

Var Pic: Pointer;
    Luar: Boolean;
    Size: Word;
    Ax,Ay: Word;
    Proc: Pointer;
    Buff1: Pointer;
    X,Y: Integer;
    Ch : Char;

Begin
  OpenVga256;
  Install_Palette('Palette.pal');
  Load_Picture('Pics3.',Screen^);
  GetMem(Pic,30000);
  GetMem(Buff1,30000);
  GetImage(Screen^,1,1,100,45,Buff1^);
  Color_Compress(Buff1^,Pic^,Size);
  FillChar(Screen^,64000,0);
  GetMem(Proc,64000);
  Luar:=False;
  X:=100;Y:=100;
  AX:=100;AY:=100;
  Repeat
    Repeat
      FillChar(Proc^,64000,1);
      Color_UnCompress(Pic^,Buff1^);
      Put3D(Proc^,X,Y,AX,AY,Buff1^);
      Move(Proc^,Screen^,64000);
    Until Keypressed;
    Ch:=Readkey;
    Case Ch of
```

```
      '2':Inc(Y,5);
       '4':Dec(X,5);
       '6':Inc(X,5);
       '8':Dec(Y,5);
       'd':Inc(AX,5);
       'a':Dec(AX,5);
       'w':Dec(AY,5);
       'x':Inc(AY,5);
       #27:Luar:=True;
     End;
  Until Luar;
  CloseVga256;
  Writeln('Ukuran Asli    : ',ImageSize(1,1,100,45));
  Writeln('Ukuran Kompresi: ',size);
  Readln;
End.
```

PENJELASAN PROGRAM

Program di atas diambil dari program di mana kita mencoba PUT dengan teknik resizing. Nah sekarang data yang diambil dengan GETIMAGE dikompres dengan menggunakan fasilitas Color_Compress dan hasilnya dimasukkan ke dalam alamat memori PIC. Demikian pula pada saat hendak dilakukannya peletakan sprite, kita harus meng-un-kompres sprite di dalam PIC ke dalam BUFF1. Setelah itu baru kita dapat meletakkan sprite tersebut. Jika Anda berusaha meletakkan sprite tersebut sebelum dibuka, gambar yang ditampilkan akan berantakan malahan bisa mengakibatkan program crashed. Tentunya saya tidak perlu menjelaskan kenapa hal ini terjadi bukan?

Bila dilihat sepintas, penggunaan sistem kompresi di atas malahan memboroskan memori banyak karena kita harus menyediakan memori untuk proses membuka kompresi sprite. Memang hal ini benar jika kita tidak memakai banyak sprite. Tetapi jika kita memakai banyak sprite, teknik ini sangat menguntungkan, karena hasil kompresi ini bisa sampai 2 atau 3 kali lebih kecil dari ukuran sprite sebenarnya. Sedangkan untuk meng-un-kompres tiap sprite tersebut kita dapat menggunakan buffer memori yang sama sehingga kita tidak perlu menyediakan buffer untuk setiap sprite. Dan yang paling penting ialah; walaupun dilakukannya proses un-kompres data setiap saat akan meletakkan sprite, animasi yang dihasilkan hampir memiliki kecepatan yang sama.

Bila Anda coba program di atas pada saat Anda menekan ESC untuk mengakhiri program, akan diberitahukan ukuran sprite aslinya dan ukuran setelah dikompres. Hasilnya cukup lumayan, hasil kompresnya memiliki ukuran 2 kali lebih kecil dari ukuran sprite sebenarnya.

KELEMAHAN SISTEM KOMPRESI SELURUH WARNA

Sebenarnya sistem kompresi ini kurang begitu sempurna. Kenapa? Jika kita memiliki data sprite yang hampir tidak pernah melakukan pengulangan titik, hasil kompresi akan berlipat ganda hingga melebihi ukuran sprite asalnya. Hal ini tentu saja tidak kita inginkan, karena tujuan kita melakukan kompresi untuk meminimumkan kebutuhan memori tidak akan tercapai malahan jadi tambah lebih besar lagi. Sebenarnya teknik pengulangan titik ini paling efisien pada saat memproses warna 0. Kenapa? Coba bayangkan sebuah sprite yang kita ambil dengan menggunakan GETIMAGE. Warna apa yang biasanya paling banyak dan umumnya berdempetan? Tentu saja warna 0 kan, karena warna ini digunakan sebagai petunjuk bahwa tidak perlu melakukan penggambaran titik. Nah untuk itulah diciptakan teknik kompresi yang lebih canggih dari COLOR COMPRESS.

TEKNIK ZERO COMPRESSION

Teknik ini dikenal dengan nama ZERO COMPRESS. Tujuannya ialah melakukan proses kompresi pada warna 0 saja. Warna selain 0 akan disimpan seperti biasanya.

Misalnya kita punya sampel data:

0 0 0 0 2 4 5 2 0 0 0 7 8 9 1

Data tersebut akan diterjemahkan dengan sistem ZERO COMPRESS sebagai berikut:

0 4 2 4 5 2 0 3 7 8 9 1

Maksud dari hasil terjemahan di atas ialah:

Warna 0 diulang 4 kali.

Warna berikutnya ialah 2, 4, 5, 2.

Warna 0 diulang 3 kali.

Warna berikutnya ialah 7, 8, 9, 1.

Untuk lebih jelasnya lihatlah listing programnya di bawah ini.

```
Procedure Zero_Compress(Var Source;Var Destination;Var Size:Word);
Var SegS,OffS :Word;
    SegD,OffD :Word;
    Length,Width:Word;
    Total     :Word;
    Finish    :Boolean;
    Value     :Byte;
    Long      :Byte;
    Packet    :Word;
    A,Ctr     :Word;
  Begin
    GetAddr(Source,SegS,OffS);
    GetAddr(Destination,SegD,OffD);
    Move(Mem[SegS:OffS],Length,2);
    Move(Mem[SegS:OffS+2],Width,2);
    Total:=Width*Length+4;
    Finish:=False;
    Value:=Mem[SegS:OffS];
    Long:=0;
    Inc(OffD,2);
    Size:=2;
    Packet:=0;
    Finish:=False;
    Ctr:=0;
    Repeat
      If Value=0 then
        Begin
          Long:=0;
          Repeat
            Inc(Long);
            Inc(OffS);
            Inc(Ctr);
          Until (Mem[SegS:OffS]<>0) or (Long>=255) or (Ctr>=Total);
          Mem[SegD:OffD]:=0;
          Mem[SegD:OffD+1]:=Long;
          Inc(OffD,2);
          Value:=Mem[SegS:OffS];
          Inc(Packet);
          Inc(Size,2);
        End else
        Begin
          Mem[SegD:OffD]:=Value;
          Inc(OffD);
          Inc(OffS);
          Value:=Mem[SegS:OffS];
          Inc(Packet);
          Inc(Ctr);
          Inc(Size);
        End;
      If Ctr>=Total then Finish:=True;
    Until Finish;
    GetAddr(Destination,SegD,OffD);
    Move(Packet,Mem[SegD:OffD],2);
  End;
```

PENJELASAN PROGRAM

Program Zero_Compress ini juga dirakit menggunakan compiler Turbo Pascal, karena dengan alasan tidak diperlukannya optimasi kecepatan. Sebenarnya cara kerja dari procedure ini sangatlah sederhana. Kita tinggal mendeteksi apakah warna yang akan kita kompres adalah warna 0. Jika ya, cari jumlah warna 0 yang berdekatan dengan posisi warna 0 sekarang ini. Kemudian rekam data dalam bentuk pasangan 2 byte yaitu angka 0 dan jumlahnya. Jika ternyata warna tersebut bukan warna 0 maka kita tinggal memasukkan nomor warna itu secara langsung ke dalam memori Destination dalam bentuk 1 byte.

```
Procedure Zero_Compress(Var Source;Var Destination;Var Size:Word);
Var SegS,OffS :Word;
    SegD,OffD :Word;
    Length,Width:Word;
    Total     :Word;
    Finish    :Boolean;
    Value     :Byte;
    Long      :Byte;
    Packet    :Word;
    A,Ctr     :Word;
  Begin
    GetAddr(Source,SegS,OffS);
    GetAddr(Destination,SegD,OffD);
```

Ambil alamat Segment dan Offset dari Source dan Destination.

```
    Move(Mem[SegS:OffS],Length,2);
    Move(Mem[SegS:OffS+2],Width,2);
    Total:=Width*Length+4;
    Finish:=False;
```

Hitung besar data yang hendak kita kompres.

```
    Value:=Mem[SegS:OffS];
    Long:=0;
    Inc(OffD,2);
    Size:=2;
```

Ambil data pertama yang akan diperiksa.
Sisakan tempat pada Destination sebesar 2 byte untuk menampung jumlah pasangan data/paket.
Set harga awal Size = 2 karena 2 byte pertama akan kita pakai untuk menyimpan jumlah paket.

```
    Packet:=0;
    Finish:=False;
    Ctr:=0;
```

Set harga awal paket = 0. Paket di sini merupakan satuan dari pasangan data yang bisa terdiri dari dua byte (bila warna 0 yang direkam), bisa juga terdiri dari satu byte (bila warna bukan 0 yang direkam).

Jadi kadang paket ini besarnya 2 byte, kadang 1 byte.

```
Repeat
  If Value=0 then
    Begin
      Long:=0;
      Repeat
        Inc(Long);
        Inc(OffS);
        Inc(Ctr);
      Until (Mem[SegS:OffS]<>0) or (Long>=255) or (Ctr>=Total);
```

Jika warna yang diproses ialah warna 0 maka hitung jumlah warna 0 yang berdekatan dan di sebelah kanan warna 0 tersebut. Jika panjang dari warna 0 yang berdekatan lebih besar dari 255 atau data sudah habis maka akhiri perhitungan panjang dari warna 0. Di sini berlaku hal yang sama dengan COLOR COMPRESS. Jika ada warna 0 sejumlah lebih dari 255 yang berdekatan, warna 0 tersebut akan disimpan dalam beberapa pasangan data.

```
      Mem[SegD:OffD]:=0;
      Mem[SegD:OffD+1]:=Long;
      Inc(OffD,2);
```

Masukkan angka 0 dan jumlahnya ke dalam Destination.

```
      Value:=Mem[SegS:OffS];
      Inc(Packet);
      Inc(Size,2);
```

Ambil data baru untuk diperiksa. Tambahkan paket karena kita baru saja mengakses sebuah paket ke dalam Destination dan tambahkan Size dengan 2 karena di sini paket berukuran 2 byte. Size ini nantinya pada akhir loop akan menunjukkan besar sprite setelah dikompresi.

```
    End else
    Begin
      Mem[SegD:OffD]:=Value;
      Inc(OffD);
      Inc(OffS);
      Value:=Mem[SegS:OffS];
      Inc(Packet);
      Inc(Ctr);
      Inc(Size);
```

Jika ternyata warna tersebut bukan warna 0 maka masukkan warna tersebut secara biasa ke dalam Destination. Jangan lupa untuk mengambil data

berikutnya dari Source untuk diperiksa selanjutnya. Kemudian tambahkan paket. Size ditambah dengan 1 karena di sini paket berukuran 1 byte.

```
        End;
      If Ctr>=Total then Finish:=True;
```

Selesai jika data telah habis.

```
    Until Finish;
```

Loop kembali.

```
    GetAddr(Destination,SegD,OffD);
    Move(Packet,Mem[SegD:OffD],2);
```

Isikan jumlah paket pada awal Destination sebagai informasi pada saat kita melakukan proses un-kompres.

```
  End;
```

Di bawah ini adalah listing program untuk membuka data yang dikompresi dengan sistem kompresi warna 0.

```
Procedure Zero_UnCompress(Var Source;Var Destination);
  Begin
  Asm
    Push ds
    Lds si,Source
    Les di,Destination
    LodsW
    Mov Cx,Ax     { Get Packed }
    @Loop1:
      Push Cx
      Lodsb
      Mov Cx,1
      Or Al,Al
      Jnz @Loop2
      Mov bl,al
      Lodsb
      Mov cl,al
      Mov al,bl
      @Loop2:
        Stosb
      Loop @Loop2
      Pop Cx
    Loop @Loop1
    Pop ds
  End;
  End;
```

PENJELASAN PROGRAM

Cara kerja dari procedure ini hampir sama dengan Color_UnCompress. Letak perbedaannya ialah pada waktu akan dilakukan perulangan data, data tersebut diperiksa apakah 0 atau tidak. Jika data tersebut 0 maka ambil byte berikutnya sebagai jumlah perulangan dari angka 0 tersebut kemudian lakukan pengisian angka 0 pada memori Destination sejumlah perulangan tersebut. Jika angka tersebut bukan 0 maka langsung saja isikan angka tersebut ke memori Destination.

```
Procedure Zero_UnCompress(Var Source;Var Destination);
  Begin
  Asm
    Push ds
    Lds si,Source
    Les di,Destination
```

Simpan DS.
Arahkan DS:SI ke Source.
Arahkan ES:DI ke Destination.

```
    LodsW
    Mov Cx,Ax     { Get Packed }
```

Ambil jumlah paket pada awal memori Destination kemudian masukkan ke CX sebagai counter pemrosesan paket.

```
    @Loop1:
      Push Cx
      Lodsb
```

Simpan CX karena CX ini nantinya akan kita gunakan sebagai counter perulangan data.
Baca data.

```
      Mov Cx,1
```

Asumsikan bahwa CX = 1. Sehingga jika data tersebut bukan 0, data tersebut cukup dimasukkan ke Destination dengan perulangan 1 kali.

```
      Or Al,Al
      Jnz @Loop2
```

Test apakah data tersebut = 0.

```
      Mov bl,al
      Lodsb
      Mov cl,al
      Mov al,bl
```

Data tersebut adalah 0 maka ambil byte berikutnya sebagai jumlah perulangan angka 0. Masukkan jumlah perulangan tadi ke counter CX menggantikan CX yang berisi harga awal 1 (karena jumlah perulangan ini tidak akan melebihi 255 maka kita cukup memasukkannya ke low register CL, CH akan selalu berisi 0).

```
@Loop2:
  Stosb
Loop @Loop2
```

Lakukan pengisian data sejumlah CX kali.

```
  Pop Cx
Loop @Loop1
```

Kembalikan CX untuk mengcounter paket.
Loop untuk memproses paket berikutnya.

```
Pop ds
```

Kembalikan DS.

```
End;
End;
```

Nah sekarang ubahlah program B_COMP1.PAS di atas, sehingga disamping menggunakan COLOR_COMPRESS dan COLOR_UNCOMPRESS juga menggunakan ZERO_COMPRESS dan ZERO_UNCOMPRESS. Segala parameter yang dikirimkan sama persis, jadi Anda cukup mengganti nama procedure-nya saja. Sebaiknya saya berikan program B_COMP1.PAS yang menggunakan ZERO_COMPRESS dan ZERO_UNCOMPRESS di bawah ini.

```
{-------------------------------------------------}
{ Program: B_COMP2.PAS                            }
{ Coder : Agustinus Nalwan                        }
{ Fungsi: Mengkompres data gambar dengan metode   }
{    Zero Compress                                }
{-------------------------------------------------}

Uses Crt,Dos,VgaCad;

Var Pic: Pointer;
    Luar: Boolean;
    Size: Word;
    Ax,Ay: Word;
    Proc: Pointer;
    Buff1: Pointer;
    X,Y: Integer;
    Ch : Char;
```

```
Begin
  OpenVga256;
  Install_Palette('Palette.pal');
  Load_Picture('Pics3.',Screen^);
  GetMem(Pic,30000);
  GetMem(Buff1,30000);
  GetImage(Screen^,1,1,100,45,Buff1^);
  Zero_Compress(Buff1^,Pic^,Size);
  FillChar(Screen^,64000,0);
  GetMem(Proc,64000);
  Luar:=False;
  X:=100;Y:=100;
  AX:=100;AY:=100;
  Repeat
    Repeat
      FillChar(Proc^,64000,1);
      Zero_UnCompress(Pic^,Buff1^);
      Put3D(Proc^,X,Y,AX,AY,Buff1^);
      Move(Proc^,Screen^,64000);
    Until Keypressed;
    Ch:=Readkey;
    Case Ch of
      '2':Inc(Y,5);
      '4':Dec(X,5);
      '6':Inc(X,5);
      '8':Dec(Y,5);
      'd':Inc(AX,5);
      'a':Dec(AX,5);
      'w':Dec(AY,5);
      'x':Inc(AY,5);
      #27:Luar:=True;
    End;
  Until Luar;
  CloseVga256;
  Writeln('Ukuran Asli    : ',ImageSize(1,1,100,45));
  Writeln('Ukuran Kompresi: ',size);
  Readln;
End.
```

PENJELASAN PROGRAM

Ya kan? Sama persis seperti program B_COMP1.PAS hanya di sini kita menggunakan ZERO_COMPRESS dan ZERO_UNCOMPRESS. Jalankan program ini dan lihatlah kualitas kompresi datanya. Anda akan melihat bahwa ukuran kompresi ZERO_COMPRESS ini sedikit lebih kecil dari COLOR_COMPRESS. Hal ini membuktikan bahwa ZERO_COMPRESS lebih handal. Tetapi ingat, kualitas kompresi dari kedua teknik ini sangat relatif. Tergantung dari pola gambar yang dikompresi. Jika gambar yang Anda kompresi banyak menggunakan warna 0 yang berdempetan,

ZERO_COMPRESS lebih unggul tetapi jika sebaliknya maka COLOR_COMPRESS yang lebih unggul. Selain itu, pola warna yang Anda gunakan juga mempengaruhi sistem kompresi ini, jika Anda sering menggunakan warna sama (bukan 0) yang berdempetan maka COLOR_COMPRESS akan lebih unggul dibandingkan dengan ZERO_COMPRESS karena warna yang berdempetan ini tidak akan dikompres oleh ZERO_COMPRESS sebagai value dan jumlahnya, tetapi langsung saja dimasukkan ke Destination satu persatu. Lain halnya dengan COLOR_COMPRESS yang akan mengkompresi seluruh warna, tidak peduli apakah warna 0 atau tidak. Sehingga pada hasil akhirnya, COLOR_COMPRESS akan lebih unggul.

TEKNIK RLL COMPRESSION

Untuk mengatasi kelemahan kedua teknik kompresi sebelumnya, maka kita perlu sebuah sistem kompresi yang dapat menyesuaikan dengan pola gambar yang akan dikompres. Untuk itulah diciptakan sistem RLL COMPRESSION. Saya sendiri tidak tahu kenapa kok disebut RLL (He he he). Teknik ini merupakan teknik yang cukup handal dan membutuhkan waktu yang cukup cepat untuk melakukan proses un-kompres. Karena itulah teknik ini dapat kita pakai untuk mengkompresi sprite selama game berlangsung.

Cara kerja dari teknik ini adalah sebagai berikut.

Jika ada warna yang berdempetan (minimal 2 titik) maka warna tersebut akan dikompres, jika tidak maka warna tersebut tidak akan dikompres. Nah bagaimana cara program un-kompres membedakan bahwa data tersebut adalah data yang dikompres atau tidak. Caranya ialah dengan menggunakan sistem paket. Paket ini ada dua macam yaitu:

- Paket yang berisi value dan jumlahnya (seperti pada teknik kompresi sebelumnya).
  Struktur datanya ialah sebagai berikut:
  - byte pertama menyimpan jumlah dari perulangan data tersebut.
  - byte kedua berisi data yang akan diulang.
- Paket berisi warna-warna yang jumlah masing-masing hanya 1 titik.
  - byte pertama menyimpan kode 0 (berfungsi sebagai kode informasi bagi program un-kompres yang menyatakan bahwa paket ini merupakan paket jenis kedua).

- byte kedua menyimpan jumlah titik dalam paket ini. (Jika byte ini berharga 0 maka menunjukkan bahwa data telah selesai).
- byte berikutnya akan menyimpan titik- titik yang tidak dikompres sebanyak jumlah yang diinstruksikan pada byte kedua.

Misalkan kita memiliki data:

1 1 1 2 4 8 2 4 5 5 5 5 5 5 5 5 5 2 3 5 6 6

Dengan sistem RLL COMPRESS data tersebut akan diubah menjadi:

3 1 0 5 2 4 8 2 4 10 5 0 3 2 3 5 2 6

Interpretasi dari data yang telah dikompres itu adalah sebagai berikut:

- 3 kali perulangan angka 1.
- Data yang tidak dikompres sejumlah 5 yaitu 2 4 8 2 4.
- 10 kali perulangan angka 5.
- Data yang tidak dikompres sejumlah 3 yaitu 2 3 5.
- 2 kali perulangan angka 6.

Untuk lebih jelasnya lihatlah program RLL_COMPRESS di bawah ini.

```
Procedure Rll_Compress(Var Source;Var Destination;Var Size:Word);
Var SegS,OffS:Word;
    SegD,OffD:Word;
    Finish:Boolean;
    SrcCtr:Byte;
    FByte:Byte;
    Ctr :Word;
    OldOFF:Word;
    Length,Width:Word;
  Begin
    Ctr:=1;
    GetAddr(Source,SegS,OffS);
    GetAddr(Destination,SegD,OffD);
    Move(Mem[SegS:OffS],Length,2);
    Move(Mem[SegS:OffS+2],Width,2);
    Size:=Length*Width+4;
    OldOFF:=OffD;
    Finish:=False;
    FByte:=Mem[SegS:OffS];
    Repeat
      SrcCtr:=1;
```

```
    While (FByte=Mem[SegS:OffS+SrcCtr]) and (SrcCtr<255) and
    (Ctr<=Size) do
      Begin
        Inc(SrcCtr);
        Inc(Ctr);
      End;
    If SrcCtr>1 then
      Begin
        Mem[SegD:OffD]:=SrcCtr;
        Inc(OffD);
        Mem[SegD:OffD]:=FByte;
        Inc(OffD);
        Inc(OffS,SrcCtr);
        FByte:=Mem[SegS:OffS];
        Inc(Ctr);
      End Else
      Begin
        Mem[SegD:OffD]:=0;
        SrcCtr:=0;
        While (Mem[SegS:OffS+SrcCtr]<>Mem[SegS:OffS+SrcCtr+1]) and
        (SrcCtr<255) and (Ctr<=Size) do
          Begin
            Inc(Ctr);
            Mem[SegD:OffD+SrcCtr+2]:=Mem[SegS:OffS+SrcCtr];
            Inc(SrcCtr);
          End;
        Mem[SegD:OffD+1]:=SrcCtr;
        Inc(OffD,SrcCtr+2);
        Inc(OffS,SrcCtr);
        FByte:=Mem[SegS:OffS];
      End;
    If Ctr>Size then Finish:=True;
  Until Finish;
  Mem[SegD:OffD]:=0;
  Mem[SegD:OffD+1]:=0;
  Size:=OffD+2-OldOFF;
End;
```

PENJELASAN PROGRAM

```
Procedure Rll_Compress(Var Source;Var Destination;Var Size:Word);
Var SegS,OffS:Word;
    SegD,OffD:Word;
    Finish:Boolean;
    SrcCtr:Byte;
    FByte:Byte;
    Ctr :Word;
    OldOFF:Word;
    Length,Width:Word;
  Begin
    Ctr:=1;
    GetAddr(Source,SegS,OffS);
    GetAddr(Destination,SegD,OffD);
```

Ambil alamat Segment dan Offset dari Source dan Destination.

```
Move(Mem[SegS:OffS],Length,2);
Move(Mem[SegS:OffS+2],Width,2);
Size:=Length*Width+4;
```

Hitung panjang data yang akan dikompres.

```
OldOFF:=OffD;
```

Simpan Offset destination awal. Nantinya harga ini digunakan untuk menentukan jumlah data setelah di-un-kompres.

```
Finish:=False;
FByte:=Mem[SegS:OffS];
```

Ambil data pertama untuk diperiksa.

```
Repeat
  SrcCtr:=1;
```

Asumsikan bahwa jumlah perulangan data yang sama adalah 1.

```
While (FByte=Mem[SegS:OffS+SrcCtr]) and (SrcCtr<255) and
(Ctr<=Size) do
  Begin
    Inc(SrcCtr);
    Inc(Ctr);
  End;
```

Jika warna berikutnya sama dengan warna yang sedang diperiksa sekarang dan jumlah perulangan data belum melebihi 255 maka tambahkan jumlah perulangan data. Di sini dilakukan proses perhitungan panjang data untuk jenis paket nomor 1.

```
If SrcCtr>1 then
  Begin
    Mem[SegD:OffD]:=SrcCtr;
    Inc(OffD);
    Mem[SegD:OffD]:=FByte;
    Inc(OffD);
    Inc(OffS,SrcCtr);
    FByte:=Mem[SegS:OffS];
    Inc(Ctr);
```

Jika jumlah perulangan data yang ditemukan minimal = 2 maka simpan data dalam bentuk paket nomor 1. Yaitu byte pertama berisi jumlahnya dan byte kedua berisi jumlahnya. Kemudian ambil data baru pada Fbyte untuk melakukan pemeriksaan data berikutnya.

```
End Else
Begin
  Mem[SegD:OffD]:=0;
  SrcCtr:=0;
```

Jika tidak maka data akan disimpan dalam bentuk paket nomor 2. Untuk itu isi byte pertama dengan kode 0.

```
While (Mem[SegS:OffS+SrcCtr]<>Mem[SegS:OffS+SrcCtr+1]) and
(SrcCtr<255) and (Ctr<=Size) do
  Begin
    Inc(Ctr);
    Mem[SegD:OffD+SrcCtr+2]:=Mem[SegS:OffS+SrcCtr];
    Inc(SrcCtr);
  End;
```

Masukkan data satu persatu hingga jumlah data telah melebihi 255 atau data telah habis atau ditemukannya data yang harus disimpan dalam bentuk paket nomor 2 (dalam hal ini ditemukannya data yang sama dan berdempetan).

```
  Mem[SegD:OffD+1]:=SrcCtr;
  Inc(OffD,SrcCtr+2);
  Inc(OffS,SrcCtr);
  FByte:=Mem[SegS:OffS];
End;
```

Jangan lupa untuk mengisi jumlah titik yang disimpan dalam paket ini pada awal dari paket. Kemudian ambil data baru ke dalam Fbyte untuk melakukan pemeriksaan data berikutnya.

```
If Ctr>Size then Finish:=True;
```

Jika data telah habis maka selesaikan loop.

```
Until Finish;
Mem[SegD:OffD]:=0;
Mem[SegD:OffD+1]:=0;
Size:=OffD+2-OldOFF;
```

Sebagai terminator/tanda henti dari data, diberi kode 0 0 (nol dua kali). Kemudian hitung Size dari hasil kompresi yaitu dengan mengurangi alamat offset destination pada saat ini dengan alamat awalnya + 2. 2 ini ditambahkan karena adanya tambahan tanda henti yang besarnya 2 byte pada akhir data.

```
End;
```

Di bawah ini adalah program yang akan kita gunakan untuk membuka data yang dikompres dengan menggunakan sistem RLL COMPRESS.

```
Procedure RLL_UnCompress (Var Source,Destination);ASSEMBLER;

Asm
  Mov dx,ds
  Les di,Destination
  Lds si,Source
  Xor bx,bx
  Cld

@1:
  Xor cx,cx
  Lodsb
  Or  al,al
  Jz  @2
  Mov cl,al
  Add bx,cx
  Lodsb
  Rep Stosb
  Loop @1

@2:
  Lodsb
  Or  al,al
  Jz  @3
  Mov cl,al
  Add bx,cx
  Rep Movsb
  Loop @1

@3:
Mov ds,dx
End;
```

PENJELASAN PROGRAM

Teknik yang digunakan untuk membuka data yang dikompres dengan sistem ini tidaklah rumit. Di sini kita tetap menggunakan compiler Assembler untuk memperoleh kecepatan maksimum. Adapun caranya adalah sebagai berikut.

Ambil byte pertama, jika byte tersebut merupakan angka 0 maka data berikut adalah jenis paket nomor 2. Langkah selanjutnya ialah mengambil byte berikutnya yang menunjukkan jumlah titik yang disimpan dalam paket ini lalu dilakukan proses pemindahan titik-titik tersebut langsung ke memori Destination. Jika ternyata angka pada byte pertama bukan 0 maka data berikut adalah jenis paket nomor 1. Untuk itu kita tinggal mengambil byte berikutnya yang berisi value/warna dari titik. Byte pertama itu sendiri menunjukkan jumlah perulangan value tersebut. Langkah selanjutnya ialah

melakukan pengisian berulang terhadap value ke memori Destination sejumlah perulangan yang diinstruksikan.

Program akan berhenti bila menemukan 2 byte berurutan yang berisi harga 0 0.

```
Procedure RLL_UnCompress (Var Source,Destination); ASSEMBLER;

Asm
  Mov dx,ds
  Les di,Destination
  Lds si,Source
  Xor bx,bx
  Cld
```

Simpan DS ke DX
Arahkan ES:DI ke Destination, DS:SI ke Source.

```
@1:
  Xor cx,cx
  Lodsb
  Or  al,al
  Jz  @2
```

Ambil data pertama, test apakah data tersebut berisi kode 0.

```
  Mov cl,al
  Add bx,cx
  Lodsb
  Rep Stosb
```

Data tersebut bukan 0 maka paket ini pasti merupakan paket jenis 1. Untuk itu masukkan harga pada byte pertama tadi ke dalam CX yang menunjukkan jumlah perulangan yang akan dilakukan. Kemudian ambil value yang sebenarnya pada byte berikutnya. Lakukan pengisian berulang sejumlah CX kali ke dalam alamat Destination.

```
  Loop @1
```

Ulang lagi ke loop untuk memeriksa data berikutnya.

```
@2:
  Lodsb
  Or  al,al
```

Data tersebut pasti merupakan paket jenis 2. Maka dari itu ambil byte berikutnya yang menunjukkan jumlah titik yang disimpan dalam paket ini. Test dahulu apakah jumlah tersebut 0, jika 0 maka data telah habis karena tanda 0 dan 0 adalah tanda henti dari data.

```
  Jz  @3
  Mov cl,al
  Add bx,cx
  Rep Movsb
  Loop @1
```

Data belum habis. Isikan jumlah titik yang akan dipindahkan ke dalam CX. Kemudian lakukan pemindahan data sebanyak CX kali. Setelah selesai, loop ke awal lagi untuk memeriksa data berikutnya.

```
@3:
  Mov ds,dx
```

Data telah habis. Kembalikan DS dari DX.

```
End;
```

Nah sekarang marilah kita mencoba kehandalan teknik RLL COMPRESSION ini dengan program yang sama seperti di atas. Kita tetap akan menggunakan sprite yang sama sehingga kita dapat membandingkan sistem kompresi yang mana yang paling efisien.

```
{-------------------------------------------------}
{ Program: B_COMP3.PAS                            }
{ Coder : Agustinus Nalwan                        }
{ Fungsi: Mengkompres data gambar dengan metode   }
{    RLL Compress                                 }
{-------------------------------------------------}

Uses Crt,Dos,VgaCad;

Var Pic: Pointer;
    Luar: Boolean;
    Size: Word;
    Ax,Ay: Word;
    Proc: Pointer;
    Buff1: Pointer;
    X,Y: Integer;
    Ch : Char;

Begin
  OpenVga256;
  Install_Palette('Palette.pal');
  Load_Picture('Pics3.',Screen^);
  GetMem(Pic,30000);
  GetMem(Buff1,30000);
  GetImage(Screen^,1,1,100,45,Buff1^);
  RLL_Compress(Buff1^,Pic^,Size);
  FillChar(Screen^,64000,0);
  GetMem(Proc,64000);
  Luar:=False;
```

```
  X:=100;Y:=100;
  AX:=100;AY:=100;
  Repeat
    Repeat
      FillChar(Proc^,64000,1);
      RLL_UnCompress(Pic^,Buff1^);
      Put3D(Proc^,X,Y,AX,AY,Buff1^);
      Move(Proc^,Screen^,64000);
    Until Keypressed;
    Ch:=Readkey;
    Case Ch of
      '2':Inc(Y,5);
      '4':Dec(X,5);
      '6':Inc(X,5);
      '8':Dec(Y,5);
      'd':Inc(AX,5);
      'a':Dec(AX,5);
      'w':Dec(AY,5);
      'x':Inc(AY,5);
      #27:Luar:=True;
    End;
  Until Luar;
  CloseVga256;
  Writeln('Ukuran Asli    : ',ImageSize(1,1,100,45));
  Writeln('Ukuran Kompresi: ',size);
  Readln;
End.
```

PENJELASAN PROGRAM

Cobalah jalankan program ini. Pada saat program selesai (tekan tombol ESC) Anda akan diberi tahu besar data setelah dikompres. Ternyata sistem kompresi ini jauh lebih efisien dibanding sistem kompresi yang telah kita pelajari sebelumnya. Mengenai masalah kecepatan un-kompres pun tidak kalah dibandingkan dengan teknik un-kompres sebelumnya. Untuk itu kita akan menggunakan teknik kompresi ini pada program-program kita mendatang.

Sebenarnya masih banyak teknik kompresi yang lain seperti kompresi secara berpola/PATTERN DATA COMPRESSION yang digunakan oleh format gambar GIF, juga ada lagi yang lebih canggih yaitu kompresi dengan menurunkan kualitas seperti yang dilakukan oleh format gambar JPEG yang bisa mengkompresi gambar 24 bit menjadi sangat kecil. Teknik-teknik tersebut sangatlah menarik dan bagus untuk dipelajari, tetapi kita tidak akan membahasnya pada kesempatan ini karena sistem un-kompres pada teknik-teknik di atas relatif jauh lebih lambat dibandingkan dengan teknik RLL COMPRESSION.

BAB 3

PEMBENAHAN KETIGA VGACAD.TPU

Pada bab ini kita akan melakukan pembenahan akhir dari VGACAD.TPU. Di sini kita akan banyak membahas mengenai penggeseran layar/background scrolling. Tentunya Anda masih ingat pada seri pertama PEMROGRAMAN ANIMASI DAN GAME PROFESIONAL kita telah membahas vertikal scrolling. Fasilitas ini masih akan kita gunakan pada VGACAD.TPU. Di samping itu kita akan menambahkan lagi dua macam fasilitas scrolling yang lebih canggih lagi yaitu horizontal scrolling dan parallax scrolling.

Horizontal scrolling ialah penggeseran layar secara horizontal/kiri-kanan. Dengan teknik scroll ini kita akan dapat membuat game seperti DOUBLE DRAGON, SUPER MARIO BROSS dan lain-lain yang menggunakan sistem scrolling ini.
Parallax scrolling ialah penggeseran layar ke segala arah, jadi tidak terikat pada penggeseran atas-bawah saja ataupun kiri-kanan saja. Dengan teknik ini kita dapat membuat game-game yang bertipe SONIC, ZONE 66 dan lain-lainnya.

Semua teknik scrolling yang akan dijelaskan pada bab ini dinamakan non-hardware scrolling di mana semuanya dilakukan dengan cara pemindahan memori. Ada teknik yang jauh lebih cepat dari teknik ini, namanya ialah hardware scrolling seperti yang digunakan pada game JACK JAZZ RABBIT. Teknik ini dilakukan dengan menggeser start awal scanning dari video card sehingga kita tidak perlu melakukan pemindahan memori sebesar layar. Cukup bagian baru yang akan dimunculkan saja yang perlu Anda letakkan pada layar, sisanya akan diatur oleh video card. Teknik ini sangat riskan untuk dipakai karena setiap video card memiliki cara yang berbeda untuk mengaksesnya. Bahkan teknik ini dapat merusakkan card yang Anda gunakan bila Anda salah dalam memberikan instruksi. Untuk itulah pada kesempatan kita kali ini, kita hanya akan membahas non-hardware scrolling saja.

PENGGESERAN LAYAR DENGAN HORIZONTAL SCROLLING

Seperti yang telah dijelaskan pada pendahuluan dari bab ini.Teknik ini hampir sama seperti vertikal scrolling, hanya arah dari scrollingnya diubah menjadi ke kiri dan ke kanan. Karena background yang akan kita pergunakan untuk scroll ini lebih besar dari 64 kilo bytes, maka kita harus menyediakan pointer lebih dari satu untuk menyimpannya. Untuk itu kita akan menggu-

nakan variabel array of pointer pada VGACAD.TPU yang sebelumnya kita gunakan untuk vertikal scrolling. Adapun struktur penyimpanannya adalah dibagi menjadi sebesar 64 kilo bytes ke arah kanan. Supaya lebih jelasnya lihatlah bagan di bawah ini.

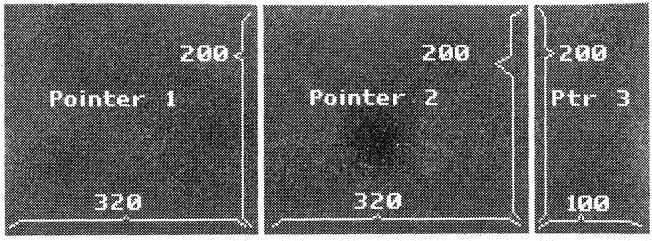


Diagram di atas menunjukkan bagaimana background yang akan kita gunakan untuk horizontal scroll disimpan dalam memori. Seperti yang telah dijelaskan di atas bahwa kita akan menggunakan beberapa variabel pointer untuk menyimpan background tersebut. Pada diagram di atas, background page dibagi menjadi 3 bagian, tiap bagiannya memiliki ukuran sebesar layar (320 x 200 titik) kecuali bagian terakhir yang ukurannya 100 x 200 titik.

Nah sekarang bagaimana cara kita menghasilkan background page tersebut? Kita dapat menggambarnya dengan menggunakan graphic editor seperti deluxe paint yang mampu mengedit gambar berukuran lebih besar dari layar, kemudian kita lakukan proses capture layar pada tiap bagiannya (pada gambar di atas kita harus melakukan capture 3 kali), lalu menggabungkan ketiga hasil capture tersebut menjadi satu file. Cara ini tidaklah efisien karena setiap kali kita hendak membuat sebuah background page, kita harus melakukan proses di atas. Untuk itu kita akan membuat sebuah program yang dapat mengkonversi file gambar dari suatu format graphic ke dalam bentuk yang siap dipakai oleh program kita nantinya.

Format graphic yang akan kita pilih ialah PCX format. Dengan program ini Anda akan dapat mengkonversi file gambar PCX beresolusi 320 x 200 dan 256 warna menjadi file SRC format. SRC format ini adalah format background page yang akan kita gunakan di VGACAD.TPU.

Untuk itu kita perlu mempelajari struktur penyimpanan gambar berformat PCX.

Format Gambar PCX

Format gambar PCX ini cukup populer di kalangan komputer graphic. Hampir semua software graphic mendukung format file ini. Oleh karena itu

format file ini saya pilih sebagai format pengantar untuk membentuk sebuah SRC file.

Sebenarnya sistem penyimpanan berformat PCX ini cukup sederhana, malahan bisa dibilang sama persis dengan sistem kompresi COLOR COMPRESS. Warna yang memiliki nomor 192 ke atas akan dikompresi dengan teknik value dan jumlah perulangan/paket RLL COMPRESS jenis 1 sedangkan warna nomor 191 ke bawah tidak akan dikompres/paket RLL COMPRESS jenis 2. Di sini kita tidak akan menggunakan kode 0 sebagai tanda pembedaan jenis paket.

Bagan sistem kompresi dari PCX file ini adalah sebagai berikut:

- Paket jenis 1. Bila data lebih besar dari 191.
- Byte pertama dikurangi 192 berisi jumlah perulangan titik pada byte berikutnya.
- Byte kedua berisi warna titik yang akan diulang.
- Paket jenis 2. Bila data lebih kecil dari 192.
- Byte pertama berisi warna titik.

Dengan kata lain, warna yang akan dikompresi adalah warna nomor 192 ke atas. Yang perlu Anda ketahui ialah pada awal dari file PCX ini selalu ada header file sebelum data gambar direkam. Untuk itu sebelum Anda melakukan pembacaan titik pertama, lompatilah terlebih dahulu header dari file PCX tersebut. Pada header file ini disimpan data-data seperti panjang dan lebar gambar, susunan palette yang digunakan, tipe kompresi dan lain-lainnya. Yang perlu kita perhatikan ialah panjang dan lebar gambar, hal-hal yang lainnya tidak begitu penting.

Perhatikan program konversi di bawah ini.

```
{-------------------------------------------------}
{ Program: PCXTO64K.PAS                           }
{ Coder : Agustinus Nalwan                        }
{ Fungsi: Konversi PCX ke Horizontal Scroll File  }
{-------------------------------------------------}

Uses Crt,Dos,VgaCad;

Type Header_Structure = Record
       Manufact: Char;
       Version: Char;
       Encoding: Char;
       BpP     : Char;
       Sx,Sy   : Integer;
```

```
        XMax,YMax: Integer;
        Hres,Vres: Integer;
        Pal      : Array[1..48] of Char;
        Reserve: Char;
        Color_Pl: Char;
        BpL      : Integer;
        Pal_Type: Integer;
        Filler : Array[1..58] of Char;
      End;

Var File_In,File_Out:File;
    Header:Header_Structure;
    St1,St2:String;
    X,Y:Integer;
    A:Integer;
    Length,Code: Byte;
    Quits:Boolean;

Begin
  Clrscr;
  St1:=ParamStr(1);
  St2:=ParamStr(2);
  Assign(File_In,St1);
  Assign(File_Out,St2);
  Reset(File_In,1);
  Rewrite(File_Out,1);
  BlockRead(File_In,Header,SizeOf(Header));
  X:=319;Y:=199;
  BlockWrite(File_Out,X,2);
  BlockWrite(File_Out,Y,2);
  X:=0;Y:=0;
  Quits:=False;
  Repeat
    BlockRead(File_In,Length,1);
    If Length>191 then
      Begin
        Length:=Length-192;
        BlockRead(File_In,Code,1);
      End Else
      Begin
        Code:=Length;
        Length:=1;
      End;
    For A:=1 to Length do
      Begin
        If X<320 then BlockWrite(File_Out,Code,1);
        Inc(X);
        If X>Header.XMax then
          Begin
            Inc(Y);
            If Y=200 then Quits:=True;
            X:=0;
          End;
      End;
```

```
  Until (IOResult<>0) or Quits;
  Close(File_In);

  Reset(File_In,1);
  BlockRead(File_In,Header,SizeOf(Header));
  X:=319;Y:=199;
  X:=Header.XMax-319;
  If X>0 then
    Begin
  BlockWrite(File_Out,X,2);
  BlockWrite(File_Out,Header.YMax,2);
  X:=0;Y:=0;
  Quits:=False;
  Repeat
  {$I-}
    BlockRead(File_In,Length,1);
  {$I+}
    If Length>191 then
      Begin
        Length:=Length-192;
 {$I-}
        BlockRead(File_In,Code,1);
 {$I+}
      End Else
      Begin
        Code:=Length;
        Length:=1;
      End;
    For A:=1 to Length do
      Begin
        If X>319 then BlockWrite(File_Out,Code,1);
        Inc(X);
        If X>Header.XMax then
          Begin
            Inc(Y);
            X:=0;
            If Y=200 then Quits:=True;
          End;
      End;
  Until (IOResult<>0) or Quits;
   End;
  Close(File_In);
  Close(File_Out);
End.
```

PENJELASAN PROGRAM

Program di atas membutuhkan dua buah parameter jika Anda jalankan. Parameter pertama ialah nama file PCX yang hendak Anda konversi dan parameter kedua ialah nama file yang akan menampung hasil kompresi.

Adapun cara kerja dari program di atas adalah dengan melakukan un-kompres terhadap gambar berformat PCX. Pada tahap pertama dilakukan perekaman titik yang ada pada baris 0 hingga 319. Caranya ialah dengan melakukan perhitungan koordinat titik yang sedang di-un-kompres. Jika titik tersebut berada pada koordinat X antara 0 hingga 319 maka rekamlah titik tersebut ke dalam file.

Tahap kedua ialah melakukan proses un-kompres ulang dari awal gambar, tetapi kali ini titik yang direkam ialah titik yang berada pada baris 320 hingga XMax.

```
Uses Crt,Dos,VgaCad;

Type Header_Structure = Record
       Manufact: Char;
       Version: Char;
       Encoding: Char;
       BpP     : Char;
       Sx,Sy   : Integer;
       XMax,YMax: Integer;
       Hres,Vres: Integer;
       Pal     : Array[1..48] of Char;
       Reserve: Char;
       Color_Pl: Char;
       BpL     : Integer;
       Pal_Type: Integer;
       Filler : Array[1..58] of Char;
     End;
```

Inilah header file dari PCX file. Dari semua ini yang akan kita pergunakan hanyalah Xmax dan Ymax yang berisi koordinat X dan Y maksimum. Kita dapat mengetahui panjang dan lebar Xmax + 1 dan Ymax + 1.

```
Var File_In,File_Out:File;
    Header:Header_Structure;
    St1,St2:String;
    X,Y:Integer;
    A:Integer;
    Length,Code: Byte;
    Quits:Boolean;

Begin
  Clrscr;
  St1:=ParamStr(1);
  St2:=ParamStr(2);
```

Ambil parameter pertama dan kedua. Parameter pertama ialah nama file PCX yang hendak dikonversi. Parameter kedua berisi nama file hasil konversi.

```
Assign(File_In,St1);
Assign(File_Out,St2);
Reset(File_In,1);
Rewrite(File_Out,1);
```

Buka file-file tersebut.

```
BlockRead(File_In,Header,SizeOf(Header));
```

Baca header file PCX.

```
X:=319;Y:=199;
BlockWrite(File_Out,X,2);
BlockWrite(File_Out,Y,2);
```

Rekam panjang dan lebar bagian background page pertama. Bagian pertama selalu memiliki panjang dan lebar yang sama dengan layar, yaitu 320 x 200 karena tidak mungkin kita menciptakan horizontal scroll yang lebih kecil dari layar.

```
X:=0;Y:=0;
Quits:=False;
Repeat
  BlockRead(File_In,Length,1);
```

Ambil data pertama.

```
If Length>191 then
  Begin
    Length:=Length-192;
    BlockRead(File_In,Code,1);
```

Jika data tersebut lebih besar dari 191 maka paket yang akan diproses adalah paket jenis 1. Kurangi data tersebut dengan 192 untuk memperoleh jumlah perulangan titik. Kemudian ambil data berikutnya sebagai warna titik yang akan diulang.

```
  End Else
  Begin
    Code:=Length;
    Length:=1;
```

Ternyata jenis paket = nomor 2. Maka data tersebut ialah warna titik itu sendiri. Set jumlah perulangan titik = 1.

```
  End;
For A:=1 to Length do
  Begin
    If X<320 then BlockWrite(File_Out,Code,1);
```

Karena kita harus menyimpan potongan-potongan gambar tersebut secara terpisah maka pertama-tama kita akan merekam gambar mulai baris 0 hingga 319.

```
Inc(X);
If X>Header.XMax then.
  Begin
    Inc(Y);
    If Y=200 then Quits:=True;
    X:=0;
  End;
```

Jika telah mencapai batas kanan gambar maka lakukan scanning lagi untuk kolom berikutnya hingga seluruh data telah terbaca.

```
    End;
Until (IOResult<>0) or Quits;
Close(File_In);
```

Loop hingga proses perekaman bagian pertama gambar telah selesai. Pada saat ini 64 kilo bytes pertama dari gambar telah direkam dalam SRC file.

```
Reset(File_In,1);
BlockRead(File_In,Header,SizeOf(Header));
X:=319;Y:=199;
X:=Header.XMax-319;
```

Buka kembali PCX file dan lompati header file. Hitung sisa baris yang ada (dengan mengurangi batas kanan gambar dengan 319). Fungsi horizontal scroll yang ada pada VGACAD.TPU ini hanya berlaku untuk gambar maksimum sebesar dua layar monitor = 640 x 200. Sebenarnya bisa saja kita membuatnya mampu untuk menangani gambar yang lebih besar dari 640 x 200 tetapi hal ini tidak banyak bermanfaat bagi kita karena untuk selanjutnya kita akan menggunakan fasilitas PARALLAX SCROLLING yang tentu saja lebih handal dari fasilitas ini. Horizontal scroll ini dibuat untuk background kecil-kecilan saja.

```
If X>0 then
  Begin
BlockWrite(File_Out,X,2);
BlockWrite(File_Out,Header.YMax,2);
```

Rekam panjang dan lebar bagian kedua dari gambar.

```
X:=0;Y:=0;
Quits:=False;
Repeat
{$I-}
  BlockRead(File_In,Length,1);
{$I+}
```

```
    If Length>191 then
      Begin
        Length:=Length-192;
{$I-}
        BlockRead(File_In,Code,1);
{$I+}
```

Baca data dengan metode paket jenis 1.

```
      End Else
      Begin
        Code:=Length;
        Length:=1;
```

Data menggunakan metode paket jenis 2.

```
      End;
    For A:=1 to Length do
      Begin
        If X>319 then BlockWrite(File_Out,Code,1);
```

Untuk kali ini kita akan merekam bagian gambar mulai kolom 320 hingga Xmaximum. Oleh karena itu di atas digunakan If X>319 yang menginstruksikan untuk melakukan penyimpanan gambar jika X lebih besar dari 319.

```
        Inc(X);
        If X>Header.XMax then
          Begin
            Inc(Y);
            X:=0;
            If Y=200 then Quits:=True;
          End;
```

Jika X telah melebihi batas kanan layar maka proses kolom berikutnya hingga seluruh data telah diproses.

```
      End;
  Until (IOResult<>0) or Quits;
   End;
  Close(File_In);
  Close(File_Out);
```

Nah sekarang bagian kedua dari background page telah disimpan dalam SRC file.

```
End.
```

Selanjutnya kita akan membahas bagaimana cara membaca background page ini ke dalam memori dan juga cara kerja dari horizontal scroll pada VGACAD.TPU.

Keempat procedure di bawah inilah yang akan memegang peranan yang penting dalam proses horizontal page.

```
Procedure Load_Horiz_Scroll (Name:String);
Var Size:Word;
    Fil:File;
  Begin
    Assign(Fil,Name);
    Reset(Fil,1);
    BlockRead(Fil,X,2);
    BlockRead(Fil,Y,2);
    GetMem(Page_Scroll[1],64000);
    BlockRead(Fil,Page_Scroll[1]^,64000);
    Px2:=0;Py2:=0;
    Px3:=0;Py3:=0;
    {$I-}
      BlockRead(Fil,X,2);
      BlockRead(Fil,Y,2);
    {$I+}
      If (IOResult=0) and (X>0) then
        Begin
          Px2:=X;Py2:=Y+1;
          Size:=(pY2)*(pX2);
          GetMem(Page_Scroll[2],Size);
          BlockRead(Fil,Page_Scroll[2]^,Size);
        End;
    Close(Fil);
  End;

Procedure Free_Horiz_Scroll;
Var Size: Word;
  Begin
    FreeMem(Page_Scroll[1],64000);
    Size:=(Py2)*(Px2);
    If Px2<>0 then
      FreeMem(Page_Scroll[2],Size);
  End;

Procedure Put_XPart(Var Picbuff,Proc;X,Start,CtrX,XGrid:Word); As-
sembler;
  Asm
    Push ds
    Les di,PicBuff
    Lds si,Proc
    Mov Cx,200
    @1:
      Push si
      Push di
      Push cx
      Add di,x
      Add si,Start
      Mov Cx,CtrX
      Rep MovsB
```

```
      Pop cx
      Pop di
      Pop si
      Add si,XGrid
      Add di,320
    Loop @1
    Pop ds
  End;

Procedure Put_Horiz_Scroll (Var PicBuff;Y:Integer);
Var S,L,T:Word;
  Begin
    If Y<320 then
      Put_XPart(PicBuff,Page_Scroll[1]^,0,Y,320-Y,320);
    If (Y<640) and (Y>0) and (Px2<>0) then
      Begin
        If Y>=320 then
          Begin
            S:=Y-320;
            L:=640-Y;
            T:=0;
          End Else
          Begin
            S:=0;
            L:=Y;
            T:=320-Y;
          End;
        Put_XPart(PicBuff,Page_Scroll[2]^,T,S,L,Px2);
      End;
  End;
```

PENJELASAN PROGRAM

PROCEDURE LOAD_HORIZ_SCROLL

Fungsi : Untuk membaca background page yang akan kita gunakan pada proses horizontal page.

Parameter: Name = String.
Nama file yang menampung background page.

Cara kerja procedure

```
Procedure Load_Horiz_Scroll (Name:String);
Var Size:Word;
    Fil:File;
  Begin
    Assign(Fil,Name);
    Reset(Fil,1);
    BlockRead(Fil,X,2);
    BlockRead(Fil,Y,2);
```

Baca panjang dan lebar bagian pertama dari gambar.

```
GetMem(Page_Scroll[1],64000);
```

Pesan memori untuk bagian pertama background page sejumlah 64000 bytes.

```
BlockRead(Fil,Page_Scroll[1]^,64000);
```

Baca bagian pertama background page sebesar 64000 bytes.

```
Px2:=0;Py2:=0;
Px3:=0;Py3:=0;
{$I-}
  BlockRead(Fil,X,2);
  BlockRead(Fil,Y,2);
{$I+}
```

Baca panjang dan lebar background page bagian kedua.

```
If (IOResult=0) and (X>0) then
  Begin
    Px2:=X;Py2:=Y+1;
    Size:=(pY2)*(pX2);
    GetMem(Page_Scroll[2],Size);
    BlockRead(Fil,Page_Scroll[2]^,Size);
```

Isikan Px2 dan Py2 yang berisi informasi panjang dan lebar bagian background page nomor 2.

Baca background page bagian kedua dan masukkan ke array pointer index nomor 2 sebesar panjang kali lebar.

```
    End;
  Close(Fil);
End;
```

PROCEDURE FREE_HORIZ_SCROLL

Fungsi : Untuk menghapus background page dari memori.
Parameter: --

Cara kerja procedure

```
Procedure Free_Horiz_Scroll;
Var Size: Word;
  Begin
    FreeMem(Page_Scroll[1],64000);
```

Hapus bagian pertama background page.

```
Size:=(Py2)*(Px2);
```

Hitung besar memori yang dipakai background page bagian kedua.

```
If Px2<>0 then
  FreeMem(Page_Scroll[2],Size);
```

Jika bagian kedua background page ada isinya, hapus bagian tersebut. Test ini bertujuan untuk memeriksa apakah background page yang kita gunakan lebih besar dari 320 baris. Jika tidak, kita tidak perlu melakukan penghapusan memori pada background page bagian kedua karena gambar sebesar 320 baris mampu ditampung oleh bagian pertama dari background page.

```
End;
```

PROCEDURE PUT_HORIZ_SCROLL

Fungsi : Menampikan horizontal scroll pada layar monitor.
Parameter: PicBuff = Pointer
Menunjukkan alamat virtual screen di mana kita akan me letakkan background page tersebut.
Y = Integer
Menunjukkan awal baris background page yang akan ditampilkan.

Cara kerja procedure

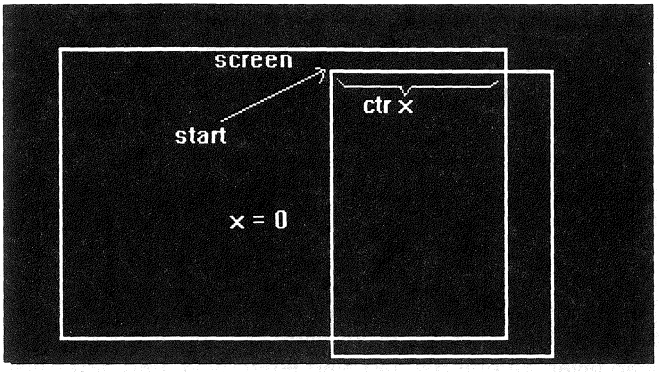
Proses peletakan horizontal page ini akan dilakukan dua kali. Pertama-tama dilakukan peletakan bagian pertama background page dengan menggunakan procedure PUT_XPART. Kemudian dilanjutkan dengan peletakan bagian kedua jika bagian kedua ini ada.

```
Procedure Put_Horiz_Scroll (Var PicBuff;Y:Integer);
Var S,L,T:Word;
  Begin
    If Y<320 then
      Put_XPart(PicBuff,Page_Scroll[1]^,0,Y,320-Y,320);
```

Jika baris awal lebih kecil dari 320 maka letakkan bagian pertama background page. Jika baris awal melebihi 319 maka bagian pertama dari background page ini tidak akan terlihat. Untuk melakukan peletakan bagian background page, kita dibantu oleh procedure PUT_XPART yang memiliki parameter sebagai berikut:

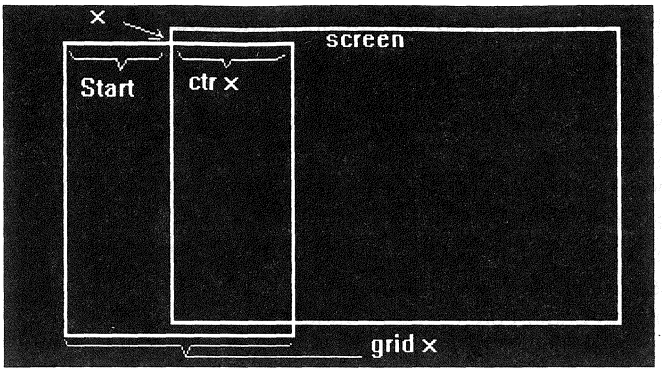
PicBuff: Virtual screen di mana background page akan diletakkan.
Proc : Pointer yang menunjukkan alamat dari bagian background page yang akan ditampilkan.
X : Koordinat X awal kita akan meletakkan bagian background page tersebut.
Start : Koordinat sprite X pertama bagian background page yang akan kita letakkan.
CtrX : Jumlah baris dari bagian background page yang akan ditampilkan.
GridX : Panjang bagian background page.

Untuk lebih jelasnya lihatlah diagram di bawah ini.



Kotak yang besar ialah layar monitor. Kotak yang kecil di sebelah kanan itu menunjukkan bagian kedua background page. Dengan diagram ini Anda akan lebih mudah mengenali parameter yang dikirimkan ke procedure PUT_XPART. Pada situasi di bawah ini, tidak ada baris dari bagian background page yang dilompati sehingga Start berharga 0. Sedangkan parameter X sendiri adalah koordinat X dari sisi kiri bagian background page. Grid X ialah panjang bagian kedua dari background page.

Pada situasi di bawah ini bagian background page pertama menembus batas kiri layar. Perhatikan parameter pada PUT_XPART.




```
If (Y<640) and (Y>0) and (Px2<>0) then
    Begin
```

Jika bagian background page yang kedua masih kelihatan pada layar monitor, lakukan proses selanjutnya.

```
If Y>=320 then
  Begin
    S:=Y-320;
    L:=640-Y;
    T:=0;
```

Perhitungan ini dilakukan bila background page akan ditampilkan dengan start awal lebih besar dari 319. Dalam hal ini akan ada bagian layar yang kosong pada sisi kanan layar, karena dengan start lebih besar dari 320 maka background page yang berukuran panjang maksimum 640 tidak akan mampu untuk menutupi seluruh layar. Oleh karena itu situasi ini tidak akan pernah terjadi jika Anda melakukan pembatasan scroll pada main program yang memanggil procedure PUT_HORIZ_PAGE ini.

```
End Else
Begin
  S:=0;
  L:=Y;
  T:=320-Y;
```

Perhitungan ini dilakukan bila background page yang akan ditampilkan memiliki start awal lebih kecil dari 320. Kondisi ini mungkin terjadi karena dengan start awal lebih kecil dari 320, background page yang memiliki panjang maksimum 640 masih mampu menutupi seluruh layar.

```
  End;
Put_XPart(PicBuff,Page_Scroll[2]^,T,S,L,Px2);
```

Letakkan background page bagian kedua pada virtual screen.

```
      End;
  End;
```

Nah sekarang marilah kita membahas procedure PUT_XPART ini.

```
Procedure Put_XPart(Var Picbuff,Proc;X,Start,CtrX,XGrid:Word);
Assembler;
  Asm
    Push ds
    Les di,PicBuff
    Lds si,Proc
```

Arahkan ES:DI ke PicBuff dan DS:SI ke Proc.

```
    Mov Cx,200
```

Isi CX dengan 200 yang berfungsi sebagai counter vertikal loop.

```
    @1:
      Push si
      Push di
      Push cx
```

Simpan SI, DI, CX dalam stack.

```
      Add di,x
```

Tambahkan DI dengan X yang merupakan start awal peletakan sisi kiri dari background page ke layar.

```
      Add si,Start
```

Tambahkan SI dengan Start yang merupakan start awal pembacaan background page.

```
      Mov Cx,CtrX
```

Isikan jumlah peletakan secara horizontal yaitu sebanyak CtrX.

```
      Rep MovsB
```

Lakukan peletakan titik sejumlah CtrX.

```
      Pop cx
      Pop di
      Pop si
```

Kembalikan CX, DI, SI dari stack.

```
    Add si,XGrid
    Add di,320
```

Tambahkan SI dengan Xgrid untuk memproses kolom selanjutnya dari bagian background page. Xgrid ini menunjukkan panjang dari bagian background page yang kita proses. Tambahkan DI dengan 320 untuk memproses kolom berikutnya pada layar monitor.

```
  Loop @1
  Pop ds
```

Loop hingga seluruh kolom pada layar monitor telah terisi. Kemudian kembalikan DS.

```
End;
```

Lumayan kan? Tidak terlalu sulit. Cobalah baca berulang-ulang jika Anda mengalami kesulitan dalam mempelajari teknik ini. Berikutnya kita akan mempelajari teknik yang lebih rumit lagi yaitu PARALLAX SCROLLING, maka dari itu sangatlah penting untuk mengerti sistem kerja horizontal dan vertikal scrolling ini sebelum mempelajari teknik PARALLAX SCROLLING.

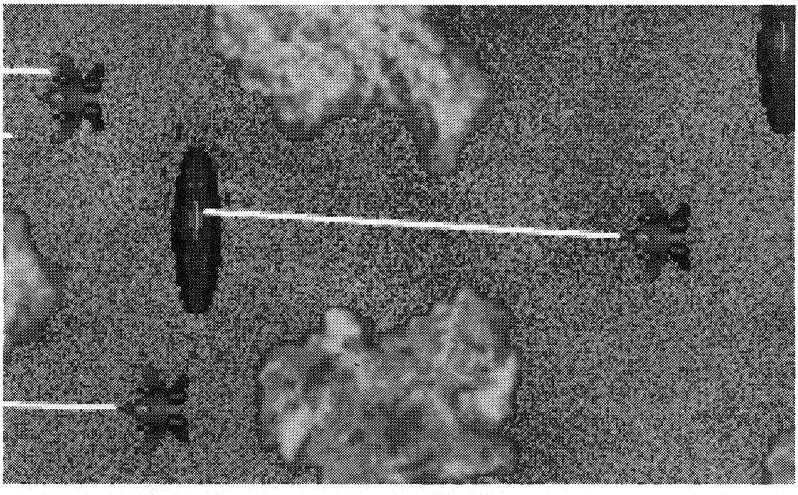
Sekarang kita akan mencoba fasilitas horizontal scroll pada VGACAD.TPU yang telah kita rancang di atas dengan sebuah program animasi sederhana.

```
{--------------------------------------}
{ Program: HORISON.PAS                 }
{ Coder : Agustinus Nalwan             }
{ Fungsi: Mencoba Horizontal Scrolling }
{--------------------------------------}

Uses Crt,Dos,VgaCad;

Var Proc,Buff:Pointer;
    Finish:Boolean;
    X,Y:Integer;
    Ch:Char;
Begin
  OpenVga256;
  X:=0;Y:=0;
  GetMem(Proc,64000);
  GetMem(Buff,64000);
  Load_Horiz_Scroll('horiz.scr');
  Install_Palette('Palette.Pal');
  Finish:=False;
  Repeat
    Repeat
      Put_Horiz_Scroll(Proc^,X);
      MoveTo(Proc^,Screen^);
    Case Ch of
      '4':Dec(X);
```

```
      '6':Inc(X);
      #27:Finish:=True;
    End;
    If X<0 then X:=0;
    If X>180 then X:=180;
    Until Keypressed;
    Ch:=Readkey;
  Until Finish;
End.
```

PENJELASAN PROGRAM

Cobalah jalankan program di atas. Anda dapat menggerakkan background yang ada di layar dengan menggunakan tombol 4 untuk menggeser ke kiri dan tombol 6 untuk menggeser ke kanan. Penggeseran akan berhenti secara otomatis bila background page telah habis.



Gambar 3.1 Output dari HORISON.PAS

Berikut akan saya jelaskan beberapa bagian penting dari program di atas.

```
Begin
  OpenVga256;
  X:=0;Y:=0;
  GetMem(Proc,64000);
```

Sediakan memori untuk animasi 3 layar.

```
  Load_Horiz_Scroll('horiz.scr');
```

Panggil procedure LOAD_HORIZ_SCROLL untuk membaca background page yang bernama HORIZ.SCR. File HORIS.SCR ini saya buat dengan menggunakan program konversi PCX kita yang lalu.

```
Install_Palette('Palette.Pal');
Finish:=False;
Repeat
  Repeat
    Put_Horiz_Scroll(Proc^,X);
```

Letakkan background page mulai kolom ke X pada virtual screen.

```
    MoveTo(Proc^,Screen^);
  Case Ch of
    '4':Dec(X);
    '6':Inc(X);
    #27:Finish:=True;
  End;
```

Deteksi tombol keyboard.

```
  If X<0 then X:=0;
  If X>180 then X:=180;
```

Deteksi apakah background page telah habis. Jika habis set harga X dengan batas maksimum/minimum.

Di sini maksimum adalah 180 karena panjang background page ialah 500 (180 + 320 = 500). Bila kita menggunakan harga X lebih dari 180 maka sebagian sisi sebelah kanan layar tidak akan tertutup oleh background page, karena background page tidak cukup. Demikian pula jika kita menggunakan X lebih kecil dari 0 (sisi sebelah kiri akan berlubang).

```
  Until Keypressed;
  Ch:=Readkey;
Until Finish;
End.
```

Cobalah untuk bermain-main dahulu dengan program di atas. Hey!! Secara tidak langsung kita telah mempelajari pembuatan background pada game-game profesional seperti SUPER MARIO BROSS, DOUBLE DRAGON dan lain -lainnya yang memanfaatkan teknik horizontal scroll. Wah Anda sudah semakin canggih sekarang. Jangan lupa masih banyak teknik-teknik canggih menanti di hadapan Anda jika Anda mengikuti terus seri PEMROGRAMAN GAME DAN ANIMASI PROFESIONAL INI.

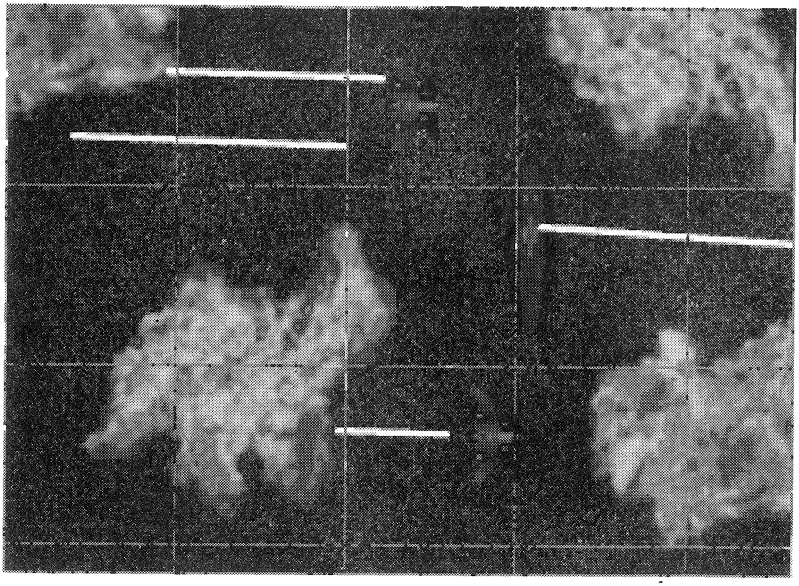
PENGGESERAN LAYAR DENGAN PARALLAX SCROLLING

Teknik ini merupakan gabungan antara teknik horizontal scrolling dan vertikal scrolling. Dengan teknik ini kita dapat melakukan scrolling ke segala arah sesuai keinginan kita. Untuk itu ukuran background page yang diperlukan juga dapat lebih bervariasi.

Karena panjang dan lebar background page bebas, maka struktur penyimpanan background page ini pada memori akan sedikit lebih rumit dibandingkan dengan sistem yang kita pakai sebelumnya pada horizontal dan vertikal scrolling.

Kita akan membagi background page yang kita pakai menjadi potongan-potongan sprite kecil berukuran 80 x 50 titik. Urutan penyimpanannya ialah seperti halnya alokasi titik pada mode $13 yaitu dari atas ke bawah dengan perulangan dari kiri ke kanan. Potongan-potongan sprite ini memiliki struktur data yang sama dengan sprite yang biasa kita ciptakan menggunakan GETIMAGE. Dengan cara ini proses parallax scrolling ini akan sedikit dipermudah.

Untuk lebih jelasnya lihatlah gambar di bawah ini.



Gambar tersebut menunjukkan bagaimana background page parallax scrolling disimpan pada memori. Masing-masing kotak berukuran 80 x 50 titik, kecuali bagian sisi kanan dan sisi bawah yang jelas berukuran lebih kecil. Tiap-tiap kotak ini adalah berupa sprite. Karena dengan fungsi parallax scrolling ini biasanya digunakan untuk background yang besar maka masing-masing bagian berukuran 80 x 50 titik itu nantinya akan dikompres pada memori, sehingga kita dapat menghemat pemakaian memori.

Wow, kedengarannya cukup rumit tapi sebenarnya tidak terlalu sulit kok.

Nah seperti halnya pada waktu kita membahas horizontal scrolling. Di sini kita perlu menciptakan sebuah program yang akan mengkonversi gambar dari format PCX ke format SCR yang merupakan data siap pakai dalam proses parallax scrolling ini. Ingat format SCR pada parallax scrolling berbeda dengan format SCR pada horizontal scrolling.

Coba pelajari program konversi berikut ini.

```
{------------------------------------ }
{ Program: PCXTOSRC.PAS                }
{ Coder : Agustinus Nalwan             }
{ Fungsi: Konversi gambar PCX ke SRC   }
{------------------------------------ }

Uses Crt,Dos,VgaCad;

Type Cell = Array[0..49,0..79] of Byte;
     Header_Structure = Record
       Manufact: Char;
       Version: Char;
       Encoding: Char;
       BpP     : Char;
       Sx,Sy   : Integer;
       XMax,YMax: Integer;
       Hres,Vres: Integer;
       Pal     : Array[1..48] of Char;
       Reserve: Char;
       Color_Pl: Char;
       BpL     : Integer;
       Pal_Type: Integer;
       Filler : Array[1..58] of Char;
     End;

Var Map    : Array[1..20,1..20] of ^Cell;
    Buffer : Pointer;
    Pack   : Pointer;
    Size   : Word;
    X,Y    : Integer;
    XMaxPage,
    YMaxPage: Integer;
    File_In: Text;
    File_In2: File;
```

```
    File_Scr: File;
    File_Atr: Text;
    Header : Header_Structure;
    St1,St2: String;
    Finish : Boolean;
    Color  : Byte;
    Ch     : Char;
    Length : Byte;
    A,B    : Integer;
    PageX,PageY,
    RX,RY  : Integer;
    XEnd,YEnd: Integer;

Begin
  St1:=ParamStr(1);
  St2:=ParamStr(2);
  For X:=1 to 10 do
    For Y:=1 to 10 do
      Begin
        GetMem(Map[Y,X],4000);
      End;
  GetMem(Buffer,10000);
  GetMem(Pack,10000);
  Assign(File_In2,St1);
  Reset(File_In2,1);
  BlockRead(File_In2,Header,SizeOf(Header));
  Close(File_In2);
  Assign(File_In,St1);
  Reset(File_In);
  For A:=1 to SizeOf(Header) do
    Read(File_In,Ch);
  Finish:=False;
  X:=0;
  Y:=0;
  Repeat
    Read(File_In,Ch);
    Color:=Ord(Ch);
    Length:=1;
    If Color>191 then
      Begin
        Length:=Color-192;
        Read(File_In,Ch);
        Color:=Ord(Ch);
      End;
    For A:=1 to Length do
      Begin
        PageX:=(X Div 80)+1;
        PageY:=(Y Div 50)+1;
        RX:=X Mod 80;
        RY:=Y Mod 50;
        Map[PageY,PageX]^[RY,RX]:=Color;
        Inc(X);
        If X>Header.XMax then
          Begin
```

```
            XMaxPage:=PageX;
            Inc(Y);
            X:=0;
            If Y>Header.YMax then
              Begin
                Finish:=True;
                YMaxPage:=PageY;
              End;
          End;
      End;
    Writeln('Processing Page: ',PageX,': ',PageY);
  Until Finish;
  Close(File_In);
  OpenVga256;
  Assign(File_Scr,St2+'.SCR');
  Rewrite(File_Scr,1);
  Assign(File_Atr,St2+'.ATR');
  Rewrite(File_Atr);
  Writeln(File_Atr,Header.XMax+1);
  Writeln(File_Atr,Header.YMax+1);
  Writeln(File_Atr,XMaxPage);
  Writeln(File_Atr,YMaxPage);
  Writeln(File_Atr,Header.XMax);
  Writeln(File_Atr,Header.YMax);
  For Y:=1 to YMaxPage do
    For X:=1 to XMaxPage do
      Begin
        If Y=YMaxPage then YEnd:=Header.YMax Mod 50 Else YEnd:=49;
        If X=XMaxPage then XEnd:=Header.XMax Mod 80 Else XEnd:=79;
        For B:=0 to YEnd do
          For A:=0 to XEnd do
            Begin
              PutPixel(Screen^,A,B,Map[Y,X]^[B,A]);
            End;
        GetImage(Screen^,0,0,XEnd,YEnd,Buffer^);
        Rll_Compress(Buffer^,Pack^,Size);
        Writeln(File_Atr,Size);
        BlockWrite(File_Scr,Pack^,Size);
      End;
  Close(File_Atr);
  Close(File_Scr);
  CloseVga256;
End.
```

PENJELASAN PROGRAM

Cara penggunaan program konversi di atas ialah dengan disertai dua buah parameter. Parameter pertama ialah nama file PCX yang hendak Anda konversi. Parameter kedua ialah nama file yang akan menampung hasil konversi (jangan diberi ekstensi). File yang dihasilkan ada dua buah. File pertama akan berekstensi ATR yang berisi:

- Baris pertama ialah panjang background page.
- Baris kedua ialah lebar background page.
- Baris ketiga ialah jumlah potongan gambar secara horizontal. Pada background page dengan situasi yang digambarkan seperti pada gambar di atas, potongan gambar secara horizontal berjumlah 5.
- Baris keempat ialah jumlah potongan gambar secara vertikal. Pada background page dengan situasi yang digambarkan seperti pada gambar di atas, potongan gambar secara vertikal berjumlah 4.
- Baris kelima tidak terpakai.
- Baris keenam tidak terpakai.
- Baris-baris selanjutnya berisi besar masing-masing potongan sprite yang telah dikompres.

File kedua berekstensi SCR berisi potongan-potongan berupa sprite yang telah dikompres.

Adapun cara kerja dari program konversi ini adalah sebagai berikut. Mula-mula kita akan membentuk variabel MAP yang merupakan sebuah array 2 dimensi of pointer. Variabel ini berukuran [20,20]. Tujuan dari variabel ini ialah sebagai penyimpanan potongan background page pada setial cell-nya. Untuk itu kita harus menyediakan memori sebesar 80 x 50 = 4000 bytes pada setiap cell dari variabel MAP. Setelah variabel ini lengkap tersedia, kita melakukan scanning secara horizontal yang diulang pada setiap kolom background page yang berada di file. Sambil melakukan scanning, kita memasukkan titik-titik yang kita temukan ke dalam cell-cell di variabel MAP.

Setelah proses scanning dilakukan, background page ini telah terbagi menjadi potongan gambar-gambar dalam tiap cell di variabel MAP.

Potongan-potongan gambar ini hanya menampung informasi warna titik-titik sedangkan dalam melakukan parallax scrolling, potongan gambar harus menampung informasi panjang dan lebar potongan gambar tersebut berikut warna titik-titik (sebagaimana halnya sprite yang dibentuk oleh GETIMAGE). Oleh sebab itu kita harus menyisipkan informasi panjang dan lebar pada awal potongan-potongan gambar. Untuk mempermudah masalah, saya lakukan proses penggambaran dari potongan gambar yang ada di tiap cell MAP ke layar kemudian dilakukan proses GETIMAGE untuk mengambil potongan gambar tersebut. Otomatis gambar yang terambil akan memiliki informasi panjang dan lebar karena dilakukannya proses GETIMAGE. Proses berikutnya ialah melakukan RLL kompresi dari setiap potongan gambar/sprite dan memasukkannya ke dalam file SCR. Jangan lupa pula me-

masukkan besar dari potongan-potongan gambar setelah dikompres pada file ATR.

```
Uses Crt,Dos,VgaCad;

Type Cell = Array[0..49,0..79] of Byte;
     Header_Structure = Record
       Manufact: Char;
       Version: Char;
       Encoding: Char;
       BpP     : Char;
       Sx,Sy   : Integer;
       XMax,YMax: Integer;
       Hres,Vres: Integer;
       Pal     : Array[1..48] of Char;
       Reserve: Char;
       Color_Pl: Char;
       BpL     : Integer;
       Pal_Type: Integer;
       Filler : Array[1..58] of Char;
     End;
```

Header dari PCX file.

```
Var Map     : Array[1..20,1..20] of ^Cell;
```

Variabel MAP yang akan kita gunakan untuk menyimpan potongan-potongan gambar.

```
    Buffer : Pointer;
    Pack   : Pointer;
    Size   : Word;
    X,Y    : Integer;
    XMaxPage,
    YMaxPage: Integer;
    File_In: Text;
    File_In2: File;
    File_Scr: File;
    File_Atr: Text;
    Header : Header_Structure;
    St1,St2: String;
    Finish : Boolean;
    Color  : Byte;
    Ch     : Char;
    Length : Byte;
    A,B    : Integer;
    PageX,PageY,
    RX,RY  : Integer;
    XEnd,YEnd: Integer;

Begin
  St1:=ParamStr(1);
  St2:=ParamStr(2);
```

Ambil nama PCX file dan nama file tujuan.

```
For X:=1 to 10 do
  For Y:=1 to 10 do
    Begin
      GetMem(Map[Y,X],4000);
    End;
```

Asumsikan besar dari PCX file yang akan diproses, maksimum ialah 10 x 10 potongan sprite, yang berarti sebesar 800 x 500 titik. Saya rasa gambar background sebesar ini sudah sangat cukup untuk membuat game dengan teknik parallax scrolling. Jika Anda ingin membuat background yang jauh lebih besar dari batas maksimum tersebut, kita tidak dapat menyimpan variabel MAP dalam memori, karena untuk ukuran 10 x 10 saja kita membutuhkan memori 10 x 10 x 4000 = 400000 bytes. Jalan satu-satunya ialah dengan menggunakan file buffer sebagai temporary memori.

```
GetMem(Buffer,10000);
GetMem(Pack,10000);
```

Sediakan memori untuk proses kompresi dan juga untuk GETIMAGE.

```
Assign(File_In2,St1);
Reset(File_In2,1);
```

Buka PCX file sebagai binary file.

```
BlockRead(File_In2,Header,SizeOf(Header));
```

Baca header PCX file.

```
Close(File_In2);
Assign(File_In,St1);
Reset(File_In);
```

Buka PCX file sebagai text file. Karena kita akan melakukan proses pembacaan satu persatu pada masing-masing data.

```
For A:=1 to SizeOf(Header) do
  Read(File_In,Ch);
```

Lompati header file PCX.

```
Finish:=False;
X:=0;
Y:=0;
Repeat
  Read(File_In,Ch);
  Color:=Ord(Ch);
  Length:=1;
```

Ambil data pertama dari PCX file. Kemudian konversikan ke dalam bilangan byte. Asumsikan panjang perulangan dari data yang akan diletakkan ialah 1.

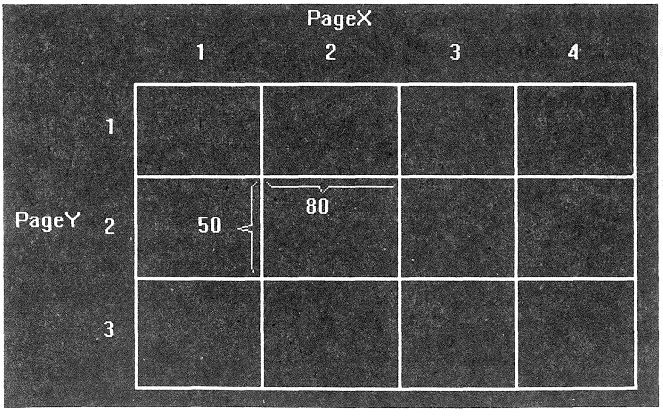
```
If Color>191 then
  Begin
    Length:=Color-192;
    Read(File_In,Ch);
    Color:=Ord(Ch);
```

Jika data yang diambil lebih besar dari 191 maka data tersebut disimpan dalam bentuk paket jenis 1 lalu kurangi data tersebut dengan 192 untuk mendapatkan panjang perulangan kemudian ambil data berikutnya sebagai data yang akan diulang.

```
  End;
For A:=1 to Length do
  Begin
    PageX:=(X Div 80)+1;
    PageY:=(Y Div 50)+1;
    RX:=X Mod 80;
    RY:=Y Mod 50;
```

Dengan koordinat titik yang sedang kita proses (X,Y) cari pada potongan gambar yang mana, titik ini harus diletakkan. Caranya ialah dengan melakukan pembagian melalui pembulatan/integer division terhadap koordinat X dengan panjang dari potongan gambar (80). Hal yang sama dilakukan juga terhadap koordinat Y yaitu membaginya dengan 50. Hasil perhitungan ini dimasukkan ke dalam variabel PAGEX dan PAGEY. Misalkan saja kita sedang memproses titik dengan koordinat (83, 56) maka koordinat page dari titik tersebut ialah (83 Div 80) +1, (56 Div 50) +1 yaitu (2,2). Untuk itu titik ini akan kita masukkan pada cell MAP [2,2].

Diagram di bawah ini akan membantu Anda untuk memperjelas proses di atas.



Langkah selanjutnya ialah mencari pada koordinat sprite berapa di dalam potongan gambar ini, titik harus diletakkan. Untuk itu kita tinggal melakukan operasi Modulus terhadap koordinat X dan Y dengan panjang dan lebar dari potongan gambar. Pada situasi di atas di mana kita mengambil titik (83,56), dengan proses Modulus, kita akan mendapatkan koordinat sprite dari titik tersebut di dalam cell MAP yaitu 83 Mod 80, 56 Mod 50 yaitu (3,6). Koordinat ini akan kita simpan dalam variabel RX dan RY. Untuk lebih jelasnya lihatlah diagram di bawah ini.

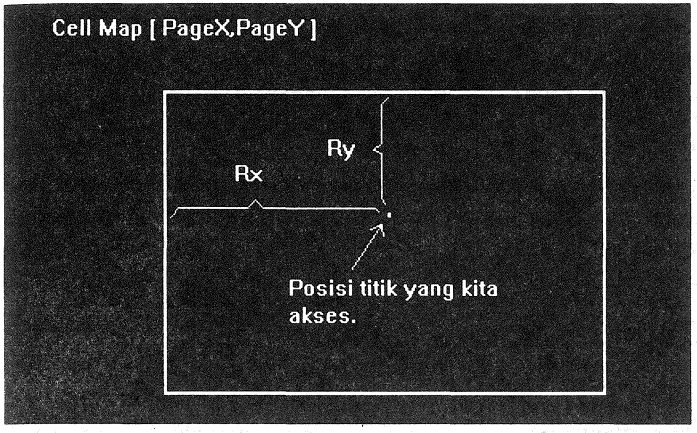


Diagram ini menggambarkan situasi di dalam sebuah cell MAP.
Rx dan Ry adalah koordinat sprite dari titik tersebut di dalam cell ini.

```
Map[PageY,PageX]^[RY,RX]:=Color;
Inc(X);
```

Masukkan titik tersebut pada lokasi yang tepat kemudian tambahkan koordinat X untuk mengakses titik berikutnya.

```
If X>Header.XMax then
  Begin
    XMaxPage:=PageX;
    Inc(Y);
    X:=0;
```

Jika koordinat X telah melebihi batas kanan background page maka tambahkan koordinat Y untuk mengakses kolom berikutnya, dan jangan lupa mengembalikan X ke 0 untuk melakukan scanning lagi dari batas kiri background page. Pada saat ini tentu saja harga dari PageX paling maksimum, karena di sini titik telah mencapai batas kanan layar. Untuk itu kita dapat mengambil harga PageX sebagai XMaxPage yang berarti PageX paling be-

sar. Tujuan dari pengambilan ini ialah sebagai informasi yang akan dituliskan pada file ATR yang akan kita bentuk.

```
If Y>Header.YMax then
  Begin
    Finish:=True;
    YMaxPage:=PageY;
  End;
```

Jika koordinat Y melebihi batas bawah layar maka keluar dari looping dan ambil harga maksimum dari PageY.

```
        End;
    End;
  Writeln('Processing Page: ',PageX,': ',PageY);
Until Finish;
Close(File_In);
OpenVga256;
Assign(File_Scr,St2+'.SCR');
Rewrite(File_Scr,1);
Assign(File_Atr,St2+'.ATR');
Rewrite(File_Atr);
```

Buka SCR dan ATR file.

```
Writeln(File_Atr,Header.XMax+1);
Writeln(File_Atr,Header.YMax+1);
Writeln(File_Atr,XMaxPage);
Writeln(File_Atr,YMaxPage);
Writeln(File_Atr,Header.XMax);
Writeln(File_Atr,Header.YMax);
```

Tuliskan informasi dari background page. Informasi ini adalah seperti yang telah kita bahas di atas (struktur file ATR).

```
For Y:=1 to YMaxPage do
  For X:=1 to XMaxPage do
    Begin
```

Ciptakan loop PageY dan PageX.

```
      If Y=YMaxPage then YEnd:=Header.YMax Mod 50 Else YEnd:=49;
      If X=XMaxPage then XEnd:=Header.XMax Mod 80 Else Xend:=79;
```

Periksa apakah PageX yang sedang kita proses adalah PageX maksimum (PageX yang paling kanan). Jika ya maka hitung panjang dari PageX ini. Karena seperti yang kita ketahui bahwa seluruh cell dari MAP ini berukuran 80 x 50 kecuali bagian kanan dan bawahnya karena besar dari background page yang kita pergunakan tidak selalu kelipatan 80 x 50.

Proses yang sama juga dilakukan terhadap PageY.

```
For B:=0 to YEnd do
  For A:=0 to XEnd do
    Begin
      PutPixel(Screen^,A,B,Map[Y,X]^[B,A]);
    End;
```

Gambar cell MAP tersebut pada layar monitor.

```
GetImage(Screen^,0,0,XEnd,YEnd,Buffer^);
```

Ambil potongan gambar tersebut dengan GETIMAGE sehingga potongan gambar sekarang berformat SPRITE (memiliki informasi panjang dan lebar pada awal datanya).

```
Rll_Compress(Buffer^,Pack^,Size);
Writeln(File_Atr,Size);
BlockWrite(File_Scr,Pack^,Size);
```

Lakukan Rll Compress dan simpan hasil kompres tersebut dalam file SCR. Jangan lupa menuliskan ukuran potongan gambar yang telah terkompresi ini dalam file ATR.

```
    End;
Close(File_Atr);
Close(File_Scr);
CloseVga256;
```

Tutup seluruh file yang kita gunakan.

```
End.
```

Sekarang saatnya kita mempelajari proses penggunaan background page untuk parallax scrolling ini. Cara pembacaan file SCR ini tidaklah terlalu rumit. Karena kita telah memiliki variabel array of pointer di dalam VGACAD.TPU yang biasanya kita gunakan untuk horizontal dan vertikal scrolling, tidak ada salahnya kalau kita memanfaatkan variabel tersebut untuk menyimpan potongan-potongan gambar dari parallax scrolling. Dengan cara ini kita tidak perlu repot-repot dalam melakukan penampilan parallax scrolling ini pada layar monitor. Kita cukup melakukan PUTIMAGE yang berulang, sehingga seluruh layar tertutup oleh background page. Penggunaan PUTIMAGE ini masuk akal, karena format dari potongan gambar adalah sama persis dengan format dari sprite yang biasa kita ciptakan dengan menggunakan fungsi GETIMAGE.

Di samping itu kita akan menciptakan sebuah record baru yang akan menampung informasi dari parallax scrolling ini. Struktur dari record tersebut adalah sebagai berikut:

- PageX yang berisi PageX maksimum.
- PageY yang berisi PageY maksimum.
- LastX tidak terpakai.
- LastY tidak terpakai.
- Length yang berisi panjang background page.
- Width yang berisi lebar background page.

Data-data di atas ini akan kita pergunakan sebagai informasi untuk background page yang sedang kita gunakan.

Sekarang lihatlah listing program yang akan kita pergunakan untuk mengakses parallax scrolling ini.

```
Procedure Load_Parallax_Scroll (Name:String);
Var No_Pic:Integer;
    File_P:File;
    File_D:Text;
    Size:Word;
    PCTR:Integer;
  Begin
    Assign(File_P,Name+'.Scr');
    Reset(File_P,1);
    Assign(File_D,Name+'.Atr');
    Reset(File_D);
    With Parallax_Dat do
      Begin
        Readln(File_D,Length);
        Readln(File_D,Width);
        Readln(File_D,Page_X);
        Readln(File_D,Page_Y);
        Readln(File_D,Last_X);
        Readln(File_D,Last_Y);
      End;
    PCTR:=1;
    Repeat
    Readln(File_D,Size);
    GetMem(Page_Scroll[PCTR],Size);
    BlockRead(File_P,Page_Scroll[PCTR]^,Size);
    Inc(PCTR);
    Until Eof(File_D);
    Close(File_P);
    Close(File_D);
  End;

Procedure Put_Parallax_Scroll(X,Y:Integer;Var PicBuff;Var Temp);
Var Al_X,Al_Y: Integer;
    Xs,Ys    : Integer;
    Xr,Yr    : Integer;
```

```
    Pic_No  : Integer;
    Size    : Word;
  Begin
    Al_X:=(X Div 80)+1;
    Al_Y:=(Y Div 50)+1;
    Xs:=(-(X Mod 80));
    Ys:=(-(Y Mod 50));
    Xr:=Xs;Yr:=Ys;
    Repeat
      Repeat
        Pic_No:=((Al_Y-1)*Parallax_Dat.Page_X)+Al_X;
        Rll_UnCompress(Page_Scroll[Pic_No]^,Temp);
        If (Xr<0) or (Xr>239) or (Yr<0) or (Yr>150)
        then ByteBlockImage(PicBuff,xr,yr,Temp) Else
          WordBlockImage(PicBuff,xr,yr,Temp);
        Inc(Xr,80);
        Inc(Al_X);
      Until Xr>319;
      Xr:=Xs;
      Al_X:=(X Div 80)+1;
      Inc(Al_Y);
      Inc(Yr,50);
    Until Yr>199;
  End;


Procedure ByteBlockImage(Var PicBuff;X1,Y1:Integer;
Var Buffer);Assembler;
Var XWStart:Word;
    XRStart:Word;
    YWStart:Word;
    YRStart:Word;
    Length:Word;
    Width:Word;
    XRead:Word;  { CX }
    YRead:Word;
  Asm
    Push ds
    Lds si,Buffer
    Les di,PicBuff
    Mov Ax,[ds:si]
    Mov Length,Ax
    Mov XRead,Ax
    Add si,2
    Mov Ax,[ds:si]
    Mov Width,Ax
    Mov YRead,Ax
    Mov YRStart,0
    Mov Ax,Y1
    Mov YWStart,Ax
    Add si,2
    Mov XRStart,0
    Mov Ax,X1
    Mov XWStart,Ax
```

```
Cmp Ax,0
Jge @X1_Not_Negative
  Neg Ax
  Mov XRStart,Ax
  Mov XWStart,0
  Sub XRead,Ax
@X1_Not_Negative:
  Mov Ax,X1
  Add Ax,Length
  Dec Ax
  Cmp Ax,319
Jle @X1_No_Right_Crop
  Sub Ax,319
  Sub XRead,Ax
@X1_No_Right_Crop:
  Mov Ax,Y1
  Cmp Ax,0
Jge @No_Top_Crop
  Neg Ax
  Mov YWStart,0
  Mov YRStart,Ax
  Sub YRead,Ax
@No_Top_Crop:
  Mov Ax,Y1
  Add Ax,Width
  Dec Ax
  Cmp Ax,199
Jle @No_Bot_Crop
  Sub Ax,199
  Sub YRead,Ax
@No_Bot_Crop:
  Add di,XWStart
  Mov Ax,YWStart
  Shl Ax,2
  Add Ax,YWStart
  Shl Ax,6
  Add di,Ax

  Mov Ax,YRStart
  Xor Ah,Ah
  Mul Length
  Add si,Ax
  Add si,XRStart

  Mov Cx,YRead
@Loop1:
  Push Cx
  Push di
  Push si
  Mov Cx,XRead
  Rep Movsb
  Pop si
  Pop di
  Add si,Length
```

```
      Add di,320
      Pop Cx
      Loop @Loop1
    Pop ds
  End;

Procedure WordBlockImage  (Var PicBuff;X,Y: Integer;
Var Buffer);Assembler;
  Asm
    Push ds             { Push Stack                          }
    Les di,PicBuff      { Arahkan ES:DI ke PICBUFF            }
    Lds si,Buffer       { Arahkan DS:SI ke BUFFER             }
    Mov bx,[ds:si]      { Baca Panjang                        }
    Inc si              { Offset + 1                          }
    Inc si              { Offset + 1                          }
    Mov cx,[ds:si]      { Baca Lebar                          }
    Inc si              { Offset + 1                          }
    Inc si              { Offset + 1                          }
    Mov dx,Y            { Masukkan Y ke Dx                    }
    Mov ax,dx           { Masukkan Dx ke Ax                   }
    Shl dx,1
    Shl dx,1            { Dx = DX * 4                         }
    Add dx,ax           { Dx = Dx + Ax                        }
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1            { Dx = Dx * 32                        }
    Add dx,X            { Dx = Dx + X                         }
    Add di,dx           { Start Offset peletakan titik        }
    Shr bx,1
    Cld                 { Counter Cx Mundur                   }
    @1:
      Mov dx,cx         { Dx = Sisa Lebar Gambar              }
      Mov cx,bx         { Cx = Sisa Panjang Gambar            }
      Push di           { Push Stack                          }
      Rep MovsW         { Baca titik dari BUFFER              }
      Pop di            { Pop Stack                           }
      Add di,320        { Register Di ke koordinat Y          }
                        { berikutnya}
      Mov cx,dx         { Isi CX dengan sisa lebar gambar     }
      Loop @1           { CX-1, Ulang hingga sisa lebar       }
                        { gambar  = 0                         }
    Pop ds              { Pop Stack                           }
  End;
```

PENJELASAN PROGRAM

Sebenarnya procedure utama yang akan kita gunakan hanyalah dua yaitu LOAD_PARALLAX_SCROLL dan PUT_PARALLAX_SCROLL. Procedure-procedure yang lain hanyalah merupakan procedure pembantu dari dua procedure tersebut.

PROCEDURE LOAD_PARALLAX_SCROLL

Fungsi : Membaca background page untuk proses parallax scrolling.
Parameter: Name = Berisi nama file background page yang akan dibaca.

Cara kerja procedure

```
Procedure Load_Parallax_Scroll (Name:String);
Var No_Pic:Integer;
    File_P:File;
    File_D:Text;
    Size:Word;
    PCTR:Integer;
  Begin
    Assign(File_P,Name+'.Scr');
    Reset(File_P,1);
    Assign(File_D,Name+'.Atr');
    Reset(File_D);
```

Buka SCR file dan ATR file.

```
    With Parallax_Dat do
      Begin
        Readln(File_D,Length);
        Readln(File_D,Width);
        Readln(File_D,Page_X);
        Readln(File_D,Page_Y);
        Readln(File_D,Last_X);
        Readln(File_D,Last_Y);
```

Masukkan informasi header file ATR ke dalam record di mana informasi parallax scroll disimpan.

```
      End;
    PCTR:=1;
    Repeat
      Readln(File_D,Size);
      GetMem(Page_Scroll[PCTR],Size);
      BlockRead(File_P,Page_Scroll[PCTR]^,Size);
```

Baca ukuran dari potongan sprite yang akan kita load. Ukuran sprite ini dapat kita peroleh dari file ATR. Kemudian lakukan pemesanan memori sebe-

sar ukuran yang kita baca tadi pada variabel array of pointer Page_Scroll dan dilanjutkan dengan membaca potongan sprite tersebut dari file SCR ke dalam variabel pointer Page_Scroll. Dengan cara ini setiap potongan sprite akan disimpan di dalam masing-masing cell dari variabel Page_Scroll.

```
Inc(PCTR);
```

Tambahkan index dari Page_Scroll.

```
Until Eof(File_D);
Close(File_P);
Close(File_D);
```

Lakukan loop sampai tidak ada lagi data yang perlu dibaca.

Tutup semua file yang kita gunakan.

```
End;
```

PROCEDURE PUT_PARALLAX_SCROLL

Fungsi : Untuk meletakkan parallax scroll pada virtual memori.

Parameter: X = Variabel integer yang menunjukkan baris awal back ground page yang hendak kita letakkan.

Y = Variabel integer yang menunjukkan kolom awal background page yang hendak kita letakkan.

PicBuff = Variabel pointer yang menunjukkan virtual screen di mana kita hendak meletakkan background page.

Temp = Variabel pointer yang menunjukkan alamat memori yang akan dipergunakan untuk melakukan Unkom pres.

Cara kerja procedure

```
Procedure Put_Parallax_Scroll(X,Y:Integer;Var PicBuff;Var Temp);
Var Al_X,Al_Y: Integer;
    Xs,Ys    : Integer;
    Xr,Yr    : Integer;
    Pic_No   : Integer;
    Size     : Word;
  Begin
    Al_X:=(X Div 80)+1;
    Al_Y:=(Y Div 50)+1;
```

Hitung PageX dan PageY awal yang akan diletakkan pada virtual screen.

```
Xs:=(-(X Mod 80));
Ys:=(-(Y Mod 50));
```

Hitung RX dan RY dari koordinat X dan Y yang diberikan. Tentunya Anda masih ingat kegunaan dari RX dan RY ini bukan? Hal ini telah kita bahas pada saat kita merancang program konversi dari PCX ke SRK. Jika Anda lupa cobalah untuk membaca kembali bagian tersebut karena kedua variabel ini memegang peranan yang sangat penting dalam proses parallax scrolling.

RX dan RY ini nantinya akan kita gunakan sebagai koordinat awal peletakan potongan sprite yang paling kiri atas.

```
Xr:=Xs;Yr:=Ys;
Repeat
  Repeat
```

Lakukan loop ganda untuk mengalokasikan potongan-potongan gambar pada virtual screen secara vertikal dan horizontal. Proses ini kita lakukan seperti pada saat kita akan meletakkan sprite pada layar monitor yaitu dengan perulangan vertikal sebagai loop terluar dan perulangan horizontal sebagai loop terdalam.

```
Pic_No:=((Al_Y-1)*Parallax_Dat.Page_X)+Al_X;
```

Karena potongan-potongan sprite tersebut kita simpan dalam variabel array 1 dimensi, maka dengan PageX dan PageY yang ada, kita berusaha untuk menerjemahkannya menjadi index bagi array 1 dimensi. Caranya sama persis dengan pada saat kita mengalokasikan titik pada virtual screen. Anggap saja koordinat X sebagai INDEX PERTAMA dan koordinat Y sebagai INDEX KEDUA dan untuk mendapatkan alamat offset, kita mengalikan INDEX KEDUA dengan 320 kemudian menambahkan hasilnya dengan INDEX PERTAMA. Di sini juga terjadi proses yang sama yaitu,
(Al_Y -1)*Parallax_Dat.Page_X + Al_X di mana:

Al_X	= PageX
Al_Y	= PageY
Parallax_Dat.Page_X	= Jumlah potongan gambar horizontal dari background page.

Al_Y dikurangi 1 karena start pertama dari PageY ialah satu maka kita harus menguranginya, sehingga PageY akan memiliki harga start = 0.

```
Rll_UnCompress(Page_Scroll[Pic_No]^,Temp);
```

Lakukan un-kompres dari data yang pada potongan gambarnya hendak kita letakkan.

```
If (Xr<0) or (Xr>239) or (Yr<0) or (Yr>150)
then ByteBlockImage(PicBuff,xr,yr,Temp) Else
  WordBlockImage(PicBuff,xr,yr,Temp);
```

Jika potongan gambar tersebut menerjang batas layar, panggil procedure BYTEBLOCKIMAGE untuk meletakkan potongan gambar tersebut. Selain itu gunakan juga procedure WORDBLOCKIMAGE.

Pada BYTEBLOCKIMAGE digunakan pengecekan batas layar, sedangkan pada WORDBLOCKIMAGE tidak. Tujuan dari pembedaan ini ialah masalah kecepatan, karena peletakan sprite tanpa pengecekan batas layar akan berjalan lebih cepat dibanding yang tidak memakai pengecekan batas layar. Maka kita tidak perlu memakai pengecekan ini jika sudah diketahui bahwa sprite tersebut tidak akan menerjang batas layar.

```
Inc(Xr,80);
Inc(Al_X);
```

Tambahkan koordinat Xr dengan 80 untuk meletakkan potongan gambar pada 80 titik berikutnya.

Tambahkan PageX untuk mengakses potongan gambar yang baru.

```
Until Xr>319;
```

Ulang hingga Xr telah melebihi batas kanan layar. Jika hal ini terjadi keluarlah dari loop, karena berarti sisa potongan gambar selanjutnya berada pada sebelah kanan batas layar. Jadi kita tidak perlu repot-repot meletakkannya.

```
Xr:=Xs;
Al_X:=(X Div 80)+1;
```

Kembalikan lagi variabel Xr ke start awal untuk melakukan scanning pada kolom berikutnya.

Kembalikan juga PageX.

```
Inc(Al_Y);
Inc(Yr,50);
```

Tambahkan PageY untuk mengakses potongan gambar pada kolom berikutnya.

Tambahkan pula Yr dengan 50 untuk meletakkan potongan gambar pada 50 titik vertikal berikutnya.

```
Until Yr>199;
```

Ulang hingga kita telah mencapai batas bawah layar.

```
End;
```

Dalam tugasnya, procedure PUT_PARALLAX_SCROLL ini dibantu oleh dua buah procedure yaitu BYTEBLOCKIMAGE dan WORDBLOCKIMAGE. Procedure ByteBlockImage ini hampir sama dengan Putimage yang menggunakan sistem pembatasan layar. Tetapi bedanya di sini kita tidak perlu mendeteksi keberadaan warna 0 karena warna 0 pun akan tetap diletakkan pada layar monitor. Sedangkan WordBlockImage, seperti namanya, proses peletakan potongan gambar di sini akan dilakukan secara per-word = 2 bytes. Karena procedure ini akan digunakan untuk meletakkan potongan gambar yang sudah pasti tidak akan melebihi batasan layar, maka kita tidak perlu melakukan proses pembatasan layar. Di samping itu, pada procedure ini juga tidak diperlukan adanya pemeriksaan terhadap warna 0. Warna 0 tetap akan ditampilkan pada layar monitor. Procedure ByteBlockImage tidak akan saya bahas di sini karena tentunya Anda dapat dengan mudah mengetahui cara kerjanya mengingat procedure ini sama persis dengan PUTIMAGE.

Di bawah ini akan dijelaskan bagaimana cara kerja procedure WordBlockImage.

Parameter dari procedure ini adalah sebagai berikut:

PicBuff = Variabel pointer yang berisi alamat virtual screen di mana kita akan meletakkan potongan sprite.
X = Koordinat X di mana kita akan meletakkan potongan gambar.
Y = Koordinat Y di mana kita akan meletakkan potongan gambar.
Buffer = Variabel pointer yang berisi alamat dari potongan gambar yang hendak kita letakkan.

```
Procedure WordBlockImage  (Var PicBuff;X,Y: Integer;
Var Buffer);Assembler;
  Asm
    Push ds             { Push Stack                  }
```

Simpan DS ke stack.

```
    Les di,PicBuff      { Arahkan ES:DI ke PICBUFF    }
    Lds si,Buffer       { Arahkan DS:SI ke BUFFER     }
```

Arahkan ES:DI dan DS:SI.

```
    Mov bx,[ds:si]      { Baca Panjang                }
    Inc si              { Offset + 1                  }
    Inc si              { Offset + 1                  }
```

```
Mov cx,[ds:si]      { Baca Lebar                    }
Inc si              { Offset + 1                    }
Inc si              { Offset + 1                    }
```

Ambil panjang dan lebar dari potongan gambar.

```
Mov dx,Y            { Masukkan Y ke Dx              }
Mov ax,dx           { Masukkan Dx ke Ax             }
Shl dx,1
Shl dx,1            { Dx = DX * 4                   }
Add dx,ax           { Dx = Dx + Ax                  }
Shl dx,1
Shl dx,1
Shl dx,1
Shl dx,1
Shl dx,1
Shl dx,1            { Dx = Dx * 32                  }
Add dx,X            { Dx = Dx + X                   }
Add di,dx           { Start Offset peletakan titik}
```

Lakukan perhitungan offset dan tambahkan ke register DI.

```
Shr bx,1
```

Bagi BX yang berisi jumlah counter yang harus dilakukan dengan 2 karena proses peletakan titik ini akan kita lakukan secara WORD sehingga counter yang diperlukan 2 kali lebih kecil dari sistem peletakan byte yang biasa kita gunakan pada PUTIMAGE sebelumnya. Ingat penggunaan peletakan bersistem WORD ini tidak dapat kita gunakan sekehendak kita. Sistem ini hanya berlaku jika panjang dari sprite yang akan kita letakkan ialah bilangan genap.

```
Cld                 { Counter Cx Mundur             }
@1:
  Mov dx,cx         { Dx = Sisa Lebar Gambar        }
  Mov cx,bx         { Cx = Sisa Panjang Gambar      }
  Push di           { Push Stack                    }
  Rep MovsW         { Baca titik dari BUFFER        }
```

Lakukan peletakan titik secara WORD.

```
    Pop di          { Pop Stack                     }
    Add di,320      { Register Di ke koordinat Y    }
                    { berikutnya                    }
    Mov cx,dx       { Isi CX dengan sisa lebar      }
                    { gambar}
    Loop @1         { CX-1, Ulang hingga sisa       }
                    { lebar gambar  = 0             }
  Pop ds            { Pop Stack                     }
End;
```

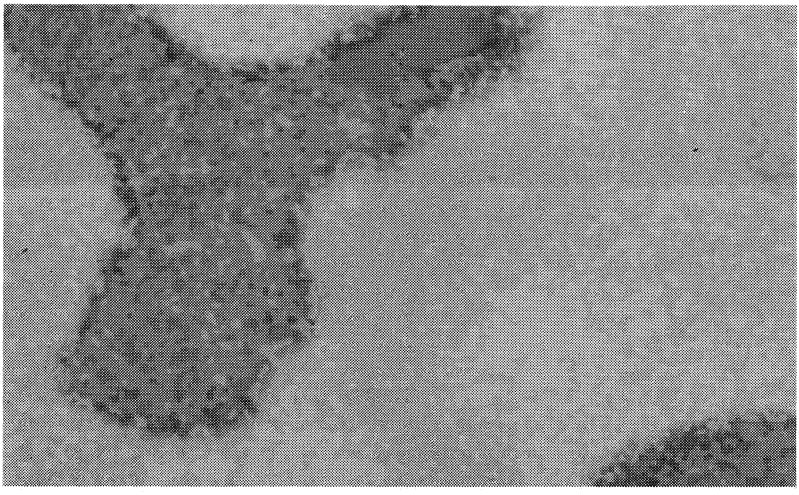
Sekarang kita akan membuat sebuah program yang akan memanfaatkan fasilitas parallax scrolling pada VGACAD.TPU. Coba perhatikan program di bawah ini.

```
{-----------------------------------------}
{ Program: PARALLAX.PAS                   }
{ Coder : Agustinus Nalwan                }
{ Fungsi: Mencoba Parallax Scrolling      }
{-----------------------------------------}

Uses Crt,Dos,VgaCad;

Var Proc,Buff:Pointer;
    Finish:Boolean;
    X,Y:Integer;
    Ch:Char;
Begin
  OpenVga256;
  X:=0;Y:=0;
  GetMem(Proc,64000);
  GetMem(Buff,64000);
  Load_Parallax_Scroll(Buff^,'parallax');
  Install_Palette('Palette.Pal');
  Finish:=False;
  Repeat
    Repeat
      Put_Parallax_Scroll(X,Y,Proc^,Buff^);
      MoveTo(Proc^,Screen^);
    Case Ch of
      '2':Inc(Y);
      '8':Dec(Y);
      '4':Dec(X);
      '6':Inc(X);
      #27:Finish:=True;
    End;
    If Y<0 then Y:=0;
    If Y>200 then Y:=200;
    If X<0 then X:=0;
    If X>280 then X:=280;
    Until Keypressed;
    Ch:=Readkey;
  Until Finish;
End.
```

PENJELASAN PROGRAM

Cobalah program ini, Anda akan melihat gambar permukaan tanah dengan bercak-bercak hijau yang maksudnya ialah hutan. Cobalah gerakkan kursor ke kiri kanan, atas bawah. WOW, rupanya teknik parallax scrolling kita telah berjalan dengan sempurna. Ingat, berilah batasan pada saat Anda melakukan scrolling. Jika Anda melakukan scrolling hingga melebihi batas back-

ground page, pada sisa layar yang kosong akan terlihat gambar kacau, karena komputer berusaha untuk menganggap data di luar alamat memori background page sebagai potongan sprite dan melakukan proses unkompress kemudian meletakkannya ke layar monitor. Hal ini dapat mengakibatkan komputer Anda crashed/macet.



Gambar 3.2 Output dari PARALLAX.PAS

Nah sampai di sinilah pembahasan kita mengenai scrolling. Pelajarilah teknik ini baik-baik karena pada akhir pembahasan buku ini, kita akan mencoba membuat sebuah game yang menggunakan teknik-teknik yang telah kita pelajari.

BAB 4

MODIFIKASI INTERUPT KEYBOARD

Pada program-program kita terdahulu, kita masih menggunakan statement READKEY dari Turbo Pascal untuk mendeteksi tombol yang kita tekan melalui keyboard. Sebenarnya statement ini sangatlah tidak tepat untuk digunakan dalam pemrograman game. Karena fungsi READKEY ini memanfaatkan interrupt keyboard dari DOS maka pada saat kita menahan sebuah tombol, akan terjadi delay sekitar 1 detik setelah terdeteksinya penekanan pertama. Dalam jangka waktu 1 detik ini READKEY akan menganggap tidak ada tombol keyboard yang ditekan walaupun kita masih tetap menahan tombol tersebut. Baru setelah selang waktu ini habis, READKEY akan menunjukkan penekanan tombol yang berulang-ulang.

Hal ini dapat Anda coba di DOS prompt. Cobalah untuk menahan huruf A. Pertama akan muncul huruf A di layar, kemudian terjadi delay 1 detik di mana tidak ada huruf yang muncul di layar. Setelah delay ini habis maka huruf A akan terus dimunculkan selama tombol A masih Anda tahan.

Sebenarnya hal ini diciptakan untuk mengurangi sensitivitas dari keyboard sehingga kita tidak akan terganggu oleh penekanan tombol yang kelebihan. Maksudnya begini, jika tidak ada delay 1 detik tersebut, Anda harus menekan setiap tombol keyboard dengan cepat pada saat mengetik. Anda tidak boleh menahan sebuah tombol lebih dari 30 ms karena komputer akan menganggap terjadinya penekanan tombol baru. Hal itu tentu saja tidak mungkin apalagi bagi orang-orang yang menggunakan word processor untuk mengetik. Bisa-bisa tulisan yang dihasilkan jadi seperti berikut "Appppaaaalllagii".

Disamping masalah tersebut, sistem pendeteksian tombol oleh DOS ini hanya dapat mendeteksi sebuah penekanan tombol keyboard. Padahal untuk game mungkin saja tiga buah tombol keyboard ditekan sekaligus. Misalnya kita hendak bergerak ke kanan atas sambil menembak, tentu saja kita harus menekan tombol kanan, atas dan tembak. Hal ini tidak mungkin dilakukan dengan menggunakan READKEY.

Bila Anda sering membuat animasi ataupun game-game yang membutuhkan penekanan tombol keyboard, Anda tentu merasa bahwa statement READKEY ini sangatlah tidak efisien. Untuk itu, pada bab ini kita akan membahas bagaimana kita mengganti interrupt DOS tersebut.

INTERRUPT $9

Sebenarnya interrupt dari DOS itu merupakan sebuah function yang dirakit dalam bahasa Assembly. Routine ini disimpan di dalam COMMAND.COM

dan akan di-load ke memori setiap saat Anda menyalakan komputer dengan sistem operasi DOS.

Adapun interrupt yang digunakan untuk membaca tombol keyboard ini ialah interrupt $9. Nah, sekarang kita akan mencoba mengganti routine interrupt $9 ini. Yang perlu Anda ingat ialah kita tidak akan mengganti routine ini secara permanent tapi hanya menggantinya untuk sementara waktu. Dan setelah kita selesai dengan program kita, routine DOS yang asli akan kita kembalikan.

Karena DOS memiliki tabel alamat untuk masing-masing interrupt-nya, kita tinggal mengganti alamat ini ke alamat procedure yang akan menggantikan interrupt $9 tersebut. Tapi ingat, sebelumnya kita harus menyimpan alamat interrupt $9 yang lama pada sebuah variabel pointer, sehingga pada akhir program kita dapat mengembalikannya lagi ke tabel alamat.

Untuk ini kita akan mulai saja dengan program sederhana yang akan mempergunakan interrupt keyboard yang telah diganti.

```
{-------------------------------------------------}
{ Program: B_KEY2.PAS                             }
{ Coder  : Agustinus Nalwan                       }
{ Fungsi : Mencoba tombol keyboard dengan         }
{          mempergunakan interrupt keyboard       }
{          buatan sendiri                         }
{-------------------------------------------------}

Uses Crt,Dos,VgaCad;

Var X,Y:Integer;
    Proc:Pointer;
    UnComp:Pointer;
    Finish:Boolean;
    OldInt9:Pointer;
    Flag,Flag2,Flag3,Flag4,Flag5,Flag6:Boolean;
    Ch:Char;
    Key:Byte;

procedure int9; interrupt; assembler;

label
  quit, release, tes2, tes3, tes5, testd, testa, testb, testc, ;
  tes4, tes6, teste;
asm
  sti
  in    al,60h
  mov   key,al
  test  al,80h
  jnz   release
  cmp   al,75
  jne   tes2
```

```
  mov   flag,1
  jmp   quit

tes2:
  cmp   al,77
  jne   tes3
  mov   flag2,1
  jmp   quit

tes3:
  cmp   al,72
  jne   tes4
  mov   flag3,1
  jmp   quit

tes4:
  cmp   al,80
  jne   tes5
  mov   flag4,1
  jmp   quit

tes5:
  cmp   al,28
  jne   tes6
  mov   flag5,1
  jmp   quit

tes6:
  cmp   al,57
  jne   quit
  mov   flag6,1
  jmp   quit

release:
  cmp   al,203
  jne   testa
  mov   flag,0
  jmp   quit

testa:
  cmp   al,205
  jne   testb
  mov   flag2,0
  jmp   quit

testb:
  cmp   al,200
  jne   testc
  mov   flag3,0
  jmp   quit

testc:
  cmp   al,208
  jne   testd
```

```
  mov   flag4,0
  jmp  quit

testd:
  cmp   al,156
  jne   teste
  mov   flag5,0
  jmp   quit

teste:
  cmp   al,185
  jne   quit
  mov   flag6,0

quit:
  mov   al,20h
  out   20h,al
end;

Procedure Initialize;
  Begin
    OpenVga256;
    Install_Palette('Palette.Pal');
    Load_Parallax_Scroll('Parallax');
    GetMem(Proc,64000);
    GetMem(UnComp,20000);
    Finish:=False;
    X:=0;Y:=0;
    GetIntVec($9,OldInt9);
    SetIntVec($9,@int9);
  End;

Procedure Main;
  Begin
    Repeat
      Put_Parallax_Scroll(X,Y,Proc^,UnComp^);
      MoveTo(Proc^,Screen^);
      If Flag4 then Inc(Y);
      If Flag3 then Dec(Y);
      If Flag then Dec(X);
      If Flag2 then Inc(X);
      If Flag5 then Finish:=True;
      If Y<0 then Y:=0;
      If Y>200 then Y:=200;
      If X<0 then X:=0;
      If X>280 then X:=280;
    Until Finish;
  End;

Procedure Removal;
  Begin
    CloseVga256;
    SetIntVec($9,OldInt9);
  End;
```

```
Begin
  Initialize;
  Main;
  Removal;
End.
```

PENJELASAN PROGRAM

Coba jalankan program di atas. WOW!! Pengontrolan keyboard kali ini jauh lebih enak dari sistem READKEY bukan? Cobalah menekan tombol bawah dan kanan sekaligus, WOW!! layar bergeser ke dua arah (kanan bawah) sekaligus. Hal ini tidak mungkin jika kita menggunakan READKEY. Juga Anda tidak akan mengalami DELAY 1 detik pada sistem penekanan tombol pada program di atas.

Untuk keluar dari program tekan tombol ENTER.

Procedure yang akan menggantikan interrupt $9 ialah procedure INT9 yang dirakit dengan bahasa assembly untuk optimasi kecepatan. Cobalah Anda perhatikan program di atas. Pada procedure initialize dilakukan penyimpanan alamat interrupt $9 yang lama ke dalam variabel pointer OLDINT9 dengan menggunakan statement GETINTVEC. Kemudian dilakukan pengisian alamat procedure INT9 ke dalam alamat interrupt $9 dengan statement SETINTVEC. Pada akhir program yaitu pada procedure Removal, dilakukan pengembalian alamat interrupt $9 yang lama.

Sekarang kita akan menelusuri program di atas satu persatu.

```
Uses Crt,Dos,VgaCad;

Var X,Y:Integer;
    Proc:Pointer;
    UnComp:Pointer;
    Finish:Boolean;
    OldInt9:Pointer;
    Flag,Flag2,Flag3,Flag4,Flag5,Flag6:Boolean;
    Ch:Char;
    Key:Byte;

procedure int9; interrupt; assembler;
```

Ini adalah procedure yang akan menggantikan interrupt $9 dari DOS. Tipenya ialah interrupt dan dirakit dengan bahasa Assembly.

```
label
  quit, release, tes2, tes3, tes5, testd, testa, testb, testc, ;
  tes4, tes6, teste;
asm
  sti
  in    al,60h
```

Baca data dari PORT[$060]. Port ini adalah port dari keyboard yang menunjukkan perubahan yang terjadi pada keyboard. Perubahan ini maksudnya ialah jika ada penekanan tombol baru ataupun adanya pelepasan tombol yang sedang ditahan. Dengan cara ini kita tidak perlu mendeteksi semua tombol yang kita gunakan melainkan tinggal melihat perubahan yang terjadi pada port ini.

Adapun data yang dikirimkan melalui port ini ialah 1 byte yang merupakan SCAN CODE dari tombol yang berubah. Untuk itu Anda perlu membuka tabel SCAN CODE karena setiap tombol keyboard memiliki SCAN CODE yang berbeda. Tabel ini biasanya dapat Anda peroleh dari buku manual DOS.

Yang perlu Anda ingat ialah; jika byte (kita sebut saja N) yang dikirimkan tersebut lebih kecil dari 128 maka tombol yang memiliki SCAN CODE N ditekan. Sedangkan jika N tersebut lebih besar atau sama dengan 128 berarti tombol yang memiliki SCAN CODE N-128 telah dilepas.

Misalnya tombol kursor atas Anda tekan maka N akan berharga 72 (SCAN CODE dari kursor atas = 72). Sementara tombol ini Anda tahan N akan berharga 0. Setelah tombol kursor atas tersebut Anda lepas maka N akan berharga 128+72 = 200. Nah dengan cara ini kita akan dapat memantau seluruh tombol yang ada di keyboard dengan mudah.

```
mov   key,al
```

Ambil sampel dari byte N yang diterima ke dalam variabel KEY.

```
test  al,80h
jz    release
```

Deteksi apakah N tersebut lebih besar atau sama dengan 128 (ingat 128 = $80). Jika ya, lakukan pendeteksian pelepasan tombol. Jika tidak, lakukan proses pendeteksian penekanan tombol.

```
cmp   al,75
jne   tes2
mov   flag,1
```

Jika bukan tombol kursor kiri ditekan (kursor kiri = SCAN CODE 75), lakukan test selanjutnya untuk mendeteksi tombol yang lain. Jika tombol kursor kiri yang ditekan, set variabel FLAG = TRUE. Dengan bantuan variabel boolean ini kita akan menyimpan kondisi dari sebuah tombol. Karena dengan kode perubahan sebuah tombol hanya akan ditampilkan sesaat pada PORT [$060] setelah itu kita tidak akan dapat mengetahui apakah tombol tersebut sedang ditahan atau tidak.

```
jmp   quit
```

Keluar dari proses, jangan mendeteksi tombol lainnya karena pada setiap kali dilakukan pembacaan PORT, hanya satu kondisi dari tombol yang akan dikirimkan.

```
tes2:
  cmp   al,77
```

Deteksi apakah tombol tersebut tombol kursor kanan (SCAN CODE = 77).

```
  jne   tes3
  mov   flag2,1
```

Jika ya, set FLAG2 = TRUE. Flag2 ini ialah variabel boolean yang akan kita gunakan untuk menyimpan kondisi dari tombol kursor kanan.

```
  jmp   quit
```

Keluar dari proses.

```
tes3:
  cmp   al,72
```

Deteksi apakah tombol tersebut ialah kursor atas.

```
  jne   tes4
  mov   flag3,1
  jmp   quit

.........
```

Demikian seterusnya. Proses yang sama akan dilakukan untuk mendeteksi tombol-tombol yang lain. Sekarang kita akan langsung membahas bagian program yang mengurusi pelepasan sebuah tombol.

```
release:
  cmp   al,203
  jne   testa
  mov   flag,0
```

Jika byte N yang diterima ialah 203 maka tombol 203-128 = 75 telah dilepas. Tombol ini ialah tombol kursor kiri. Maka dari itu variabel FLAG yang menyimpan kondisi tombol kursor kiri diset menjadi FALSE. Jika ternyata byte N tersebut bukan 203 maka lakukan test untuk tombol lainnya.

```
  jmp  quit
```

Keluar dari proses, karena telah ditemukan bahwa byte N yang dikirimkan menunjukkan bahwa tombol kursor kiri dilepas. Sehingga tidak perlu dilakukan pendeteksian tombol lainnya.

```
testa:
  cmp   al,205
  jne   testb
  mov   flag2,0
```

Deteksi apakah tombol yang dilepas ialah 205-128 = 77 yaitu tombol kursor kanan. Jika ya maka set FLAG2 menjadi FALSE.

```
  jmp  quit
```

Keluar dari proses.

```
.....
```

Demikian seterusnya hingga pada akhir program.

```
quit:
  mov   al,20h
  out   20h,al
```

Kirimkan harga $20 ke PORT[$020] yang memerintahkan keyboard untuk melakukan pendeteksian byte N baru pada PORT[$060]. Jika tidak dilakukannya fungsi PORT[$020]:=$20 tersebut di atas, harga pada PORT[$060] tidak akan diupdate/diubah untuk mendeteksi kondisi tombol yang baru.

```
end;
```

Nah routine di ataslah yang akan menggantikan fungsi dari interrupt $9 kita. Dengan funsi yang baru ini kita tidak akan dapat menggunakan tombol CTRL ALT DEL lagi untuk melakukan boot karena perintah CTRL ALT DEL tersebut ada pada fungsi interrupt $9 yang lama. Untuk itu berhati-hatilah dalam membuat program yang menggunakan interrupt $ 9 buatan kita sendiri. Jangan lupa untuk memberikan tombol yang akan kita gunakan sebagai terminator program. Jika tidak, Anda tidak akan dapat keluar dari program Anda sendiri dan terpaksa harus mereset komputer. Yang lebih mengerikan lagi jika Anda belum menyimpan program tersebut ke disk. Anda tidak akan dapat kembali ke editor karena tombol CTRL BREAK juga tidak berfungsi.

```
Procedure Initialize;
  Begin
    OpenVga256;
    Install_Palette('Palette.Pal');
    Load_Parallax_Scroll('Parallax');
    GetMem(Proc,64000);
    GetMem(UnComp,20000);
    Finish:=False;
    X:=0;Y:=0;
    GetIntVec($9,OldInt9);
```

Baca alamat interrupt $9 yang lama dan simpan ke dalam variabel pointer OldInt9.

```
    SetIntVec($9,@int9);
```

Isikan alamat interrupt $9 dengan alamat dari procedure INT9 buatan kita.

```
  End;

Procedure Main;
  Begin
    Repeat
      Put_Parallax_Scroll(X,Y,Proc^,UnComp^);
      MoveTo(Proc^,Screen^);
      If Flag4 then Inc(Y);
      If Flag3 then Dec(Y);
      If Flag then Dec(X);
      If Flag2 then Inc(X);
```

Karena kita menggunakan variabel FLAG, FLAG2, FLAG3, FLAG4, FLAG5 untuk menyimpan kondisi tombol kursor kiri, kursor kanan, kursor atas, kursor bawah dan tombol ENTER. Oleh karena itu kita tinggal mendeteksi, apakah variabel tersebut TRUE yang berarti tombol yang bersangkutan sedang ditahan, ataukah FALSE yang berarti tombol yang bersangkutan sedang dilepas.

```
      If Flag5 then Finish:=True;
```

Jika tombol ENTER ditekan, keluarlah dari program.

```
      If Y<0 then Y:=0;
      If Y>200 then Y:=200;
      If X<0 then X:=0;
      If X>280 then X:=280;
```

Lakukan pembatasan dari parallax scrolling.

```
    Until Finish;
  End;

Procedure Removal;
  Begin
    CloseVga256;
    SetIntVec($9,OldInt9);
```

Kembalikan alamat interrupt $9 yang lama.

```
  End;

Begin
  Initialize;
```

```
  Main;
  Removal;
End.
```

Untuk menunjukkan beda dari sistem interrupt $9 yang baru dengan yang lama, cobalah program di bawah ini. Program ini sama persis seperti program di atas, tetapi masih menggunakan sistem READKEY sebagai pembacaan tombol keyboard..

```
{-----------------------------------------------------}
{ Program: B_KEY1.PAS                                 }
{ Coder  : Agustinus Nalwan                           }
{ Fungsi : Mencoba tombol keyboard dengan             }
{          mempergunakan interrupt keyboard           }
{          standard.                                  }
{-----------------------------------------------------}

Uses Crt,Dos,VgaCad;

Var X,Y:Integer;
    Proc:Pointer;
    UnComp:Pointer;
    Finish:Boolean;
    Ch:Char;

Procedure Initialize;
  Begin
    OpenVga256;
    Install_Palette('Palette.Pal');
    Load_Parallax_Scroll('Parallax');
    GetMem(Proc,64000);
    GetMem(UnComp,20000);
    Finish:=False;
    X:=0;Y:=0;
  End;

Procedure Main;
  Begin
    Repeat
      Repeat
        Put_Parallax_Scroll(X,Y,Proc^,UnComp^);
        MoveTo(Proc^,Screen^);
      Until Keypressed;
      Ch:=Readkey;
    Case Ch of
      '2':Inc(Y);
      '8':Dec(Y);
      '4':Dec(X);
      '6':Inc(X);
      #27:Finish:=True;
    End;
    If Y<0 then Y:=0;
```

```
    If Y>200 then Y:=200;
    If X<0 then X:=0;
    If X>280 then X:=280;
    Until Finish;
  End;

Procedure Removal;
  Begin
  End;

Begin
  Initialize;
  Main;
  Removal;
End.
```

PENJELASAN PROGRAM

Coba jalankan program tersebut. Jelek sekali kan!! Coba tahan tombol 6. Terjadi delay 1 detik setelah pembacaan pertama kan? Juga scroll yang dihasilkan jadi tersendat-sendat. Sekarang coba tekan tombol 6 dan 2 untuk melakukan scroll kanan bawah. Tidak bisa kan? Sistem READKEY ini tidak dapat mendeteksi penekanan dua buah tombol sekaligus.

Nah mulai saat ini kita tidak akan menggunakan sistem READKEY lagi untuk membuat program-program kita selanjutnya, kecuali pada saat-saat tertentu yang tidak diperlukannya pendeteksian tombol pada saat animasi berlangsung.

BAB 5

TEKNIK MOMENTUM DAN TABRAKAN

Pada bab ini kita akan membahas hal-hal yang berkenaan dengan kecepatan, percepatan, momentum, dan collision. Rasanya kok seperti pelajaran fisika aja nih. Bagi pembaca yang sedang duduk di SMA kelas 2 tentunya topik ini tidak asing bukan. Apalagi bagi yang mengambil jurusan Fisika. Memang dunia pemrograman game itu tidak dapat lepas dari matematika maupun fisika. Karena dengan kedua ilmu ini kita akan dapat memvisualisasikan dunia yang sebenarnya di dalam game yang akan kita buat. Tentu saja kedua ilmu yang diterapkan dalam pembuatan game ini jauh lebih menarik, jadi jangan khawatir.

TEKNIK GERAKAN MOMENTUM

Pada bab-bab terdahulu, maupun pada buku seri pertama, semua objek yang kita gerakkan baik dalam game maupun dalam animasi memiliki tipe gerakan dengan kecepatan linear. Maksudnya ialah benda-benda tersebut bergerak dengan kecepatan konstan. Padahal sebenarnya jika kita membayangkan akan membuat game yang bertipe SUPER MARIO BROSS di mana Anda dapat menggerakkan si MARIO ke kanan. Si MARIO akan mulai bergerak dengan kecepatan pelan, kecepatan ini akan terus bertambah hingga akhirnya ia akan bergerak dengan kecepatan maksimum. Demikian pula jika si MARIO melakukan pengereman (habis berlari). Dia tidak akan langsung berhenti melainkan berhenti secara perlahan-lahan. Inilah yang disebut dengan teknik gerakan momentum.

Kenapa kok disebut momentum? Karena di sini hukum kekekalan momentum berlaku, yaitu momentum dari MARIO yang berlari yaitu - Kecepatan * Massa - sebagian akan ditransfer ke momentum gesekan/friksion dengan permukaan tanah.

Sudah deh, kita tidak akan membahas masalah fisika ini secara jauh. Pokoknya yang kita inginkan ialah kecepatan tidak boleh naik tiba-tiba tapi harus perlahan-lahan, demikian pula jika ingin berhenti harus perlahan-lahan. Tidak boleh tiba-tiba.

Nah sekarang bagaimana cara kita menghasilkan teknik gerakan momentum ini? Gampang kok. Pada program-program kita sebelumnya, ingat tidak waktu kita membuat FIRE FIGHTER? Posisi pesawat ditambah dengan suatu harga yang konstan. Sehingga pesawat akan bergerak dengan kecepatan konstan pula. Nah sekarang untuk menghasilkan gerakan momentum, kita akan menambah posisi pesawat dengan sebuah variabel yang berubah-ubah harganya. Variabel ini kita namakan V. V ini akan berkurang terus sehingga

menjadi 0 jika tidak dilakukan penekanan tombol. Tapi jika misalnya kita menekan tombol kanan, V ini akan ditambah terus hingga tombol kanan dilepas, maka secara otomatis posisi pesawat akan ditambah dengan harga yang akan selalu bertambah. Dengan cara ini pesawat akan bergerak secara momentum. Jika kita lepas tombol kanan tersebut maka V akan dikurangi perlahan-lahan menjadi 0. Dengan cara ini pesawat akan terlihat bergerak dengan kecepatan yang makin pelan dan akhirnya berhenti pada saat V = 0.

Cobalah program di bawah ini dan rasakan momentum yang dihasilkan.

```
{-------------------------------------------------}
{ Program: B_MOM3.PAS                             }
{ Coder : Agustinus Nalwan                        }
{ Fungsi: Mencoba menggerakkan sprite dengan      }
{           sistem momentum movement.             }
{-------------------------------------------------}

Uses Crt,Dos,VgaCad;

Var X,Y:Integer;
    Proc:Pointer;
    SpdX:Integer;
    Pic:Pointer;
    Finish:Boolean;
    OldInt9:Pointer;
    Flag,Flag2,Flag3,Flag4,Flag5,Flag6:Boolean;
    Ch:Char;
    Key:Byte;

procedure int9; interrupt; assembler;

label
  quit, release, tes2, tes3, tes5, testd, testa, testb, testc, ;
  tes4, tes6, teste;
asm
  sti
  in    al,60h
  mov   key,al
  test  al,80h
  jnz   release
  cmp   al,75
  jne   tes2
  mov   flag,1
  jmp   quit

tes2:
  cmp   al,77
  jne   tes3
  mov   flag2,1
  jmp   quit
```

```
tes3:
  cmp   al,72
  jne   tes4
  mov   flag3,1
  jmp   quit

tes4:
  cmp   al,80
  jne   tes5
  mov   flag4,1
  jmp   quit

tes5:
  cmp   al,28
  jne   tes6
  mov   flag5,1
  jmp   quit

tes6:
  cmp   al,57
  jne   quit
  mov   flag6,1
  jmp   quit

release:
  cmp   al,203
  jne   testa
  mov   flag,0
  jmp   quit

testa:
  cmp   al,205
  jne   testb
  mov   flag2,0
  jmp   quit

testb:
  cmp   al,200
  jne   testc
  mov   flag3,0
  jmp   quit

testc:
  cmp   al,208
  jne   testd
  mov   flag4,0
  jmp   quit

testd:
  cmp   al,156
  jne   teste
  mov   flag5,0
  jmp   quit
```

```
teste:
  cmp   al,185
  jne   quit
  mov   flag6,0

quit:
  mov   al,20h
  out   20h,al
end;

Procedure Initialize;
  Begin
    OpenVga256;
    Install_Palette('Palette.Pal');
    GetMem(Proc,64000);
    GetMem(Pic,20000);
    Finish:=False;
    X:=160;Y:=150;
    SpdX:=0;
    GetIntVec($9,OldInt9);
    SetIntVec($9,@int9);
  End;

Procedure Main;
  Begin
    Repeat
      FillChar(Proc^,64000,0);
      PutImage(Proc^,X,Y,Pic^);
      MoveTo(Proc^,Screen^);
      If Flag then
        If SpdX>-8 then Dec(SpdX);
      If Flag2 then
        If SpdX<8 then Inc(SpdX);
      Inc(X,SpdX);
      If Not Flag And Not Flag2 then
        Begin
          If SpdX>0 then Dec(SpdX);
          If SpdX<0 then Inc(SpdX);
        End;
      If Flag5 then Finish:=True;
      If X<0 then X:=0;
      If X>280 then X:=280;
      Delay(20);
    Until Finish;
  End;

Procedure Removal;
  Begin
    CloseVga256;
    SetIntVec($9,OldInt9);
  End;

Procedure Generate_Pic;
Var A:Integer;
```

```
  Begin
    FillChar(Screen^,64000,0);
    For A:=1 to 20 do
      Circle(Screen^,160,100,A,A+16);
    GetImage(Screen^,140,80,180,120,Pic^);
  End;

Begin
  Initialize;
  Generate_Pic;
  Main;
  Removal;
End.
```

PENJELASAN PROGRAM

Jika Anda jalankan program di atas, pada layar monitor akan ditampilkan sebuah bola. Sekarang cobalah gerakkan bola tersebut ke kiri dan kanan dengan menggunakan kursor. Bola tersebut akan bergerak dengan menggunakan momentum. Terasa kan bedanya ?

Kita tidak akan membahas seluruh bagian dari program di atas, karena tentunya Anda telah mengerti sebagian besar dari program tersebut. Procedure yang akan dibahas ialah procedure MAIN yang memvisualisasikan sistem momentum ini.

```
Procedure Main;
  Begin
    Repeat
      FillChar(Proc^,64000,0);
      PutImage(Proc^,X,Y,Pic^);
      MoveTo(Proc^,Screen^);
      If Flag then
        If SpdX>-8 then Dec(SpdX);
```

Jika tombol kiri ditekan, kurangi kecepatan (SpdX). Tetapi SpdX ini tidak boleh lebih kecil dari -8 yang berarti maksimum kecepatan ke arah kiri ialah 8. Jika kita tidak melakukan pembatasan terhadap SpdX ini, SpdX akan terus berkurang bila kita tetap menahan tombol kiri, padahal kita tidak ingin objek yang akan kita gerakkan memiliki kecepatan yang tak terhingga kan? Kecepatan maksimum ke kiri sebesar 8 ini dinamakan terminal velocity/kecepatan maksimum yang dapat ditempuh oleh sebuah benda. Terminal velocity ini dapat Anda bayangkan pada saat Anda start berlari, mula-mula kecepatan Anda pelan kemudian makin lama makin cepat hingga pada batas tertentu kecepatan Anda akan tetap konstan tidak bertambah.

```
If Flag2 then
    If SpdX<8 then Inc(SpdX);
```

Jika tombol kanan yang ditekan maka tambah kecepatan. Sistem terminal velocity juga berlaku di sini yaitu pada kecepatan maksimum ke arah kanan sebesar 8.

```
Inc(X,SpdX);
```

Tambahkan posisi objek dengan kecepatannya pada saat itu.

```
If Not Flag And Not Flag2 then
  Begin
    If SpdX>0 then Dec(SpdX);
    If SpdX<0 then Inc(SpdX);
  End;
```

Jika tidak ada tombol yang ditekan maka kurangi kecepatan ke arah 0. Pada saat ini objek melakukan pengereman.

```
If Flag5 then Finish:=True;
```

Tombol ENTER untuk keluar dari program.

```
If X<0 then X:=0;
If X>280 then X:=280;
```

Batasi posisi objek. Jangan sampai melebihi batas layar.

```
Delay(20);
```

Delay untuk mengurangi kecepatan animasi.

```
  Until Finish;
End;
```

Dengan teknik momentum ini, gerakan benda akan menjadi lebih lembut karena gerakan ini seperti halnya gerakan normal yang terjadi pada dunia nyata.

Teknik momentum ini tidak hanya digunakan untuk menggerakkan sprite saja. Malahan sistem scrolling pun bisa juga menerapkan teknik ini. Dengan ini scrolling yang dihasilkan akan lebih lembut dan kadang bisa juga menipu penglihatan kita sehingga scrolling yang dihasilkan terasa bertambah cepat.

Kalau tidak percaya cobalah program di bawah ini.

```
{-------------------------------------------------}
{ Program: B_MOM2.PAS                             }
{ Coder  : Agustinus Nalwan                       }
{ Fungsi: Mencoba Scroll dengan sistem momentum   }
{         movement.                               }
{-------------------------------------------------}

Uses Crt,Dos,VgaCad;

Var X,Y:Integer;
    Proc:Pointer;
    SpdX,SpdY:Integer;
    UnComp:Pointer;
    Finish:Boolean;
    OldInt9:Pointer;
    Flag,Flag2,Flag3,Flag4,Flag5,Flag6:Boolean;
    Ch:Char;
    Key:Byte;

procedure int9; interrupt; assembler;

label
  quit, release, tes2, tes3, tes5, testd, testa, testb, testc, ;
  tes4, tes6, teste;
asm
  sti
  in    al,60h
  mov   key,al
  test  al,80h
  jnz   release
  cmp   al,75
  jne   tes2
  mov   flag,1
  jmp   quit

tes2:
  cmp   al,77
  jne   tes3
  mov   flag2,1
  jmp   quit

tes3:
  cmp   al,72
  jne   tes4
  mov   flag3,1
  jmp   quit

tes4:
  cmp   al,80
  jne   tes5
  mov   flag4,1
  jmp   quit
```

```
tes5:
  cmp   al,28
  jne   tes6
  mov   flag5,1
  jmp   quit

tes6:
  cmp   al,57
  jne   quit
  mov   flag6,1
  jmp   quit

release:
  cmp   al,203
  jne   testa
  mov   flag,0
  jmp   quit

testa:
  cmp   al,205
  jne   testb
  mov   flag2,0
  jmp   quit

testb:
  cmp   al,200
  jne   testc
  mov   flag3,0
  jmp   quit

testc:
  cmp   al,208
  jne   testd
  mov   flag4,0
  jmp   quit

testd:
  cmp   al,156
  jne   teste
  mov   flag5,0
  jmp   quit

teste:
  cmp   al,185
  jne   quit
  mov   flag6,0

quit:
  mov   al,20h
  out   20h,al
end;
```

```
Procedure Initialize;
  Begin
    OpenVga256;
    Install_Palette('Palette.Pal');
    Load_Parallax_Scroll('Back1');
    GetMem(Proc,64000);
    GetMem(UnComp,20000);
    Finish:=False;
    X:=0;Y:=0;
    SpdX:=0;SpdY:=0;
    GetIntVec($9,OldInt9);
    SetIntVec($9,@int9);
  End;

Procedure Main;
  Begin
    Repeat
      Put_Parallax_Scroll(X,Y,Proc^,UnComp^);
      MoveTo(Proc^,Screen^);
      If Flag4 then
        If SpdY<8 then Inc(SpdY);
      If Flag3 then
        If SpdY>-8 then Dec(SpdY);
      If Flag then
        If SpdX>-8 then Dec(SpdX);
      If Flag2 then
        If SpdX<8 then Inc(SpdX);
      Inc(X,SpdX);
      Inc(Y,SpdY);
      If Not Flag And Not Flag2 then
        Begin
          If SpdX>0 then Dec(SpdX);
          If SpdX<0 then Inc(SpdX);
        End;
      If Not Flag3 And Not Flag4 then
        Begin
          If SpdY>0 then Dec(SpdY);
          If SpdY<0 then Inc(SpdY);
        End;
      If Flag5 then Finish:=True;
      If Y<0 then Y:=0;
      If Y>300 then Y:=300;
      If X<0 then X:=0;
      If X>2680 then X:=2680;
    Until Finish;
  End;

Procedure Removal;
  Begin
    CloseVga256;
    SetIntVec($9,OldInt9);
  End;
```

```
Begin
  Initialize;
  Main;
  Removal;
End.
```

PENJELASAN PROGRAM

Pada saat Anda menjalankan program ini cobalah untuk menggerakkan parallax page ke kiri-kanan-atas-bawah. Scroll yang dihasilkan lembut kan? Malahan terlihat cepat sekali. Pada bagian ini kita juga akan membahas main programnya saja, karena bagian program yang lain sangatlah sederhana dan telah dijelaskan beberapa kali pada bab-bab sebelumnya.

```
Procedure Main;
  Begin
    Repeat
      Put_Parallax_Scroll(X,Y,Proc^,UnComp^);
```

Tampilkan parallax page dengan koordinat sprite X,Y.

```
      MoveTo(Proc^,Screen^);
      If Flag4 then
        If SpdY<8 then Inc(SpdY);
```

Jika tombol bawah ditekan, maka jika kecepatan sumbu Y belum mencapai terminal velocity 8, tambahkan kecepatan sumbu Y.

Ingat pada program kita kali ini, kita menggunakan dua variabel SPEED yaitu SpdX dan SpdY. SpdX adalah V untuk sumbu X sedangkan SpdY adalah V untuk sumbu Y. Karena kali ini kita akan menggerakkan layar ke arah atas-bawah dan kiri-kanan. Pada program kita yang lalu kita cukup menggunakan SpdX karena kita hanya akan menggerakkan bola ke kiri-kanan saja.

Secara fisika hal ini benar, karena momentum sumbu X dan momentum sumbu Y dapat dihitung secara terpisah (fisika lagi ..... fisika lagi).

```
      If Flag3 then
        If SpdY>-8 then Dec(SpdY);
```

Jika tombol atas ditekan, maka jika kecepatan sumbu Y belum mencapai terminal velocity -8, kurangi kecepatan sumbu Y.

```
      If Flag then
        If SpdX>-8 then Dec(SpdX);
```

Jika tombol kiri ditekan, maka jika kecepatan sumbu X belum mencapai terminal velocity -8, kurangi kecepatan sumbu X.

```
If Flag2 then
  If SpdX<8 then Inc(SpdX);
```

Jika tombol kanan ditekan, maka jika kecepatan sumbu X belum mencapai terminal velocity 8, tambahkan kecepatan sumbu X.

```
Inc(X,SpdX);
Inc(Y,SpdY);
```

Tambahkan posisi X dan Y sprite dari layar dengan SpdX dan SpdY.

```
If Not Flag And Not Flag2 then
  Begin
    If SpdX>0 then Dec(SpdX);
    If SpdX<0 then Inc(SpdX);
  End;
```

Seperti pada program kita terdahulu, jika tidak ditekannya tombol kiri ataupun kanan maka perlahan-lahan SpdX harus diatur mendekati 0. Sehingga dilakukan pengereman terhadap sumbu X.

```
If Not Flag3 And Not Flag4 then
  Begin
    If SpdY>0 then Dec(SpdY);
    If SpdY<0 then Inc(SpdY);
  End;
```

Dilakukan pengereman terhadap sumbu Y jika tombol atas dan tombol bawah tidak ditekan.

```
If Flag5 then Finish:=True;
```

Akhiri program jika tombol ENTER ditekan.

```
If Y<0 then Y:=0;
If Y>300 then Y:=300;
If X<0 then X:=0;
If X>2680 then X:=2680;
```

Batasi gerakan parallax scrolling. WOW!! gambar yang discroll bagus ya?? Gambar inilah yang nantinya akan kita gunakan sebagai parallax background page dari permainan yang akan kita rancang pada bab terakhir. Tunggu tanggal mainnya!!

```
  Until Finish;
End;
```

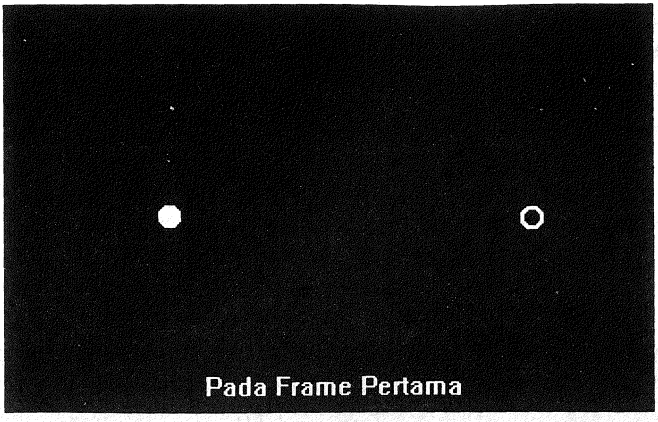
Coba sekarang Anda jalankan lagi program kita terdahulu yaitu B_KEY2.PAS. Pada program tersebut kita masih menggunakan teknik linear movement. Bandingkan dengan program B_MOM2.PAS ini.

Teknik ini digunakan pada game-game seperti JACK JAZZ RABBIT lho. Secara perlahan dan pasti kita telah menjajaki taraf yang masih merupakan impian pada saat kita memulai buku seri pertama SAP PEMROGRAMAN ANIMASI DAN GAME PROFESIONAL.

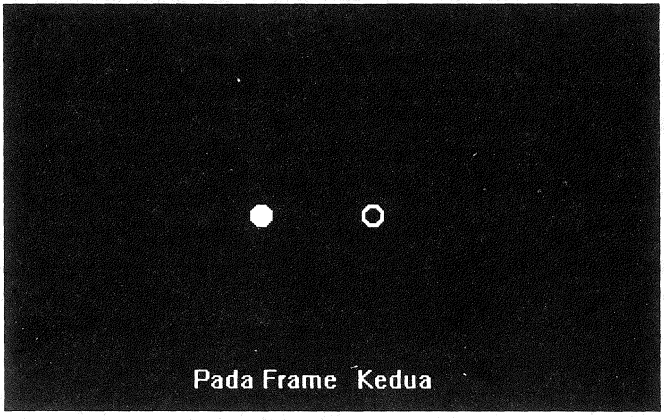
TEKNIK TABRAKAN DENGAN PENDETEKSIAN PIXEL

Jika kita berkecimpung dalam bidang gerakan antarobjek, mau tidak mau kita juga harus mempelajari teknik tabrakan/collision antara objek. Pada buku seri pertama SAP PEMROGRAMAN ANIMASI DAN GAME PROFESIONAL kita baru mempelajari kulit luar dari teknik tabrakan antara dua benda. Tentunya Anda masih ingat pada saat kita membuat FIRE FIGHTER, kita mempergunakan sistem pendeteksian dua buah persegi panjang (teknik ini dinamakan teknik pendeteksian pixel) yang mewakili dua buah objek yang sedang bergerak. Sebenarnya teknik perhitungan tabrakan ini tidak semudah itu. Kita tidak cukup hanya mendeteksi posisi suatu benda pada setiap kali benda tersebut berpindah tempat. Kenapa?

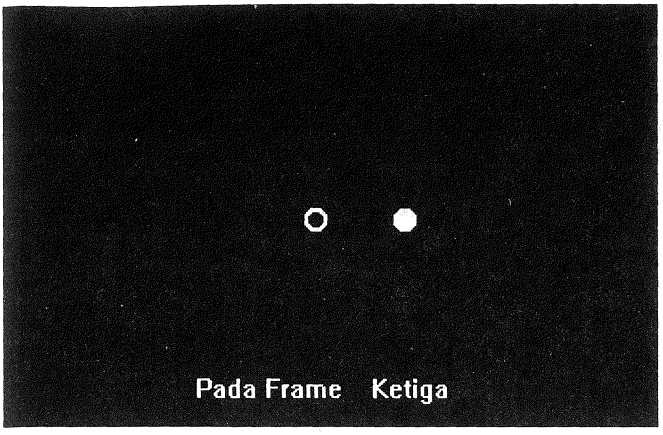
Coba bayangkan kita membuat program di mana ada dua buah bola bergerak dengan kecepatan yang sangat tinggi. Bola pertama bergerak dari arah kiri ke kanan dengan kecepatan 50 pixel per frame, sedangkan bola kedua bergerak dari arah kanan ke kiri dengan kecepatan 50 pixel per frame. Posisi kedua bola tersebut pada koordinat Y yang sama sehingga secara teori kedua bola ini harusnya bertabrakan. Karena besar dari kedua bola ini hanya 2 x 2 pixel maka ada kemungkinan tabrakan dua bola ini tidak terdeteksi oleh program. Untuk lebih jelasnya lihatlah diagram berikut.



Ini adalah situasi frame pertama yang menggambarkan bola A dan bola B yang hendak bertabrakan. Di sini dengan teknik perpotongan dua buah persegi panjang yang mengelilingi masing-masing objek, tidak terdeteksi adanya tabrakan.



Sekarang pada frame kedua, masih belum juga terdeteksi adanya tabrakan. Bola A masih bergerak dengan cepat ke arah kanan sedangkan bola B bergerak ke arah kiri.



Sekarang situasi pada frame ketiga. OO!! Kedua bola telah melewati tempat tabrakan sehingga pada frame ini tabrakan tidak terdeteksi oleh program. Seharusnya tabrakan telah terjadi di antara frame 2 dan frame 3 ini, tetapi program hanya dapat mendeteksi adanya tabrakan pada setiap adanya pergantian frame saja. Memang untuk objek yang berukuran besar dan berkecepatan rendah, hal ini jarang terjadi karena dengan besarnya ukuran objek maka perpotongan antara kedua objek akan lebih mudah terjadi apalagi kalau didukung dengan kecepatan objek yang rendah.

Situasi di atas dapat disimulasikan pada program di bawah ini.

```
{-------------------------------------------------}
{ Program: B_COL1.PAS                             }
{ Coder : Agustinus Nalwan                        }
{ Fungsi: Mencoba sistem deteksi tabrakan antara  }
{         2 objek dengan sistem pixel             }
{-------------------------------------------------}

Uses Crt,Dos,VgaCad;

Type Block_Rec = Record
       X1,Y1,X2,Y2:Integer;
     End;

Var X,Y:Integer;
    Block:Block_Rec;
    Proc:Pointer;
    SpdX:Integer;
    Pic:Pointer;
    Finish:Boolean;
    OldInt9:Pointer;
    Flag,Flag2,Flag3,Flag4,Flag5,Flag6:Boolean;
    Ch:Char;
    Key:Byte;
```

```
procedure int9; interrupt; assembler;

label
  quit, release, tes2, tes3, tes5, testd, testa, testb, testc, ;
  tes4, tes6, teste;
asm
  sti
  in    al,60h
  mov   key,al
  test  al,80h
  jnz   release
  cmp   al,75
  jne   tes2
  mov   flag,1
  jmp   quit

tes2:
  cmp   al,77
  jne   tes3
  mov   flag2,1
  jmp   quit

tes3:
  cmp   al,72
  jne   tes4
  mov   flag3,1
  jmp   quit

tes4:
  cmp   al,80
  jne   tes5
  mov   flag4,1
  jmp   quit

tes5:
  cmp   al,28
  jne   tes6
  mov   flag5,1
  jmp   quit

tes6:
  cmp   al,57
  jne   quit
  mov   flag6,1
  jmp   quit

release:
  cmp   al,203
  jne   testa
  mov   flag,0
  jmp   quit
```

```
testa:
  cmp   al,205
  jne   testb
  mov   flag2,0
  jmp   quit

testb:
  cmp   al,200
  jne   testc
  mov   flag3,0
  jmp   quit

testc:
  cmp   al,208
  jne   testd
  mov   flag4,0
  jmp   quit

testd:
  cmp   al,156
  jne   teste
  mov   flag5,0
  jmp   quit

teste:
  cmp   al,185
  jne   quit
  mov   flag6,0

quit:
  mov   al,20h
  out   20h,al
end;


Procedure Initialize;
  Begin
    OpenVga256;
    Install_Palette('Palette.Pal');
    GetMem(Proc,64000);
    GetMem(Pic,20000);
    Finish:=False;
    X:=160;Y:=150;
    SpdX:=0;
    GetIntVec($9,OldInt9);
    SetIntVec($9,@int9);
  End;
Function Check_Collision(X,Vx:Integer):Boolean;
Var Rx:Integer;
  Begin
    Rx:=X+Vx;
    Check_Collision:=False;
    With Block do
      Begin
```

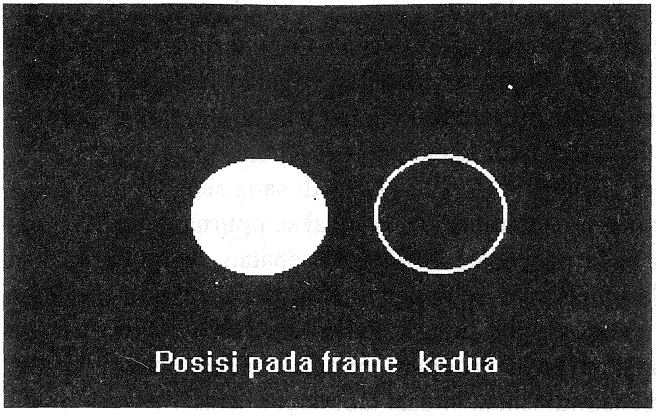
```
        If (Rx+10>=X1) and (Rx<=X2) then Check_Collision:=True;
      End;
  End;

Procedure Main;
  Begin
    Repeat
      FillChar(Proc^,64000,0);
      PutImage(Proc^,X,Y,Pic^);
      VgaCad.Block(Proc^,Block.X1,Block.Y1,Block.X2,Block.Y2,15);
      MoveTo(Proc^,Screen^);
      If Flag then
        If Check_Collision(X,-40)=False then Dec(X,40);
      If Flag2 then
        If Check_Collision(X,40)=False then Inc(X,40);
      If Flag5 then Finish:=True;
      If X<0 then X:=0;
      If X>280 then X:=280;
      Delay(20);
    Until Finish;
  End;

Procedure Removal;
  Begin
    CloseVga256;
    SetIntVec($9,OldInt9);
  End;

Procedure Generate_Pic;
Var A:Integer;
  Begin
    FillChar(Screen^,64000,0);
    For A:=1 to 5 do
      Circle(Screen^,160,100,A,A+16);
    GetImage(Screen^,155,95,165,105,Pic^);
  End;

Procedure Generate_Data;
  Begin
    With Block do
      Begin
        X1:=60;
        Y1:=60;
        X2:=62;
        Y2:=180;
      End;
  End;
Begin
  Initialize;
  Generate_Pic;
  Generate_Data;
  Main;
  Removal;
End.
```

PENJELASAN PROGRAM

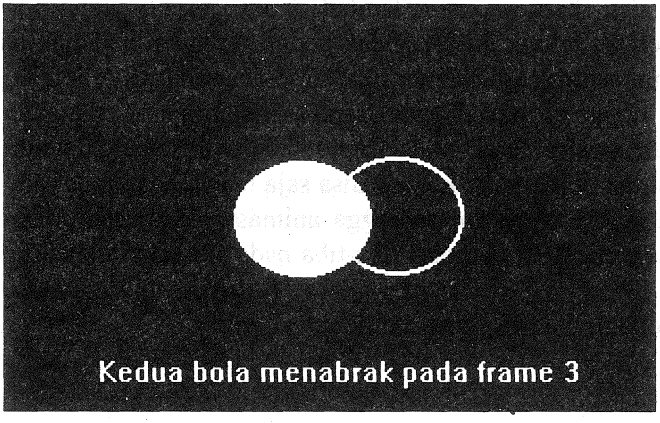
Cobalah untuk menjalankan program di atas dan gerakkan bola dengan cepat ke kiri dan ke kanan sehingga menerjang dinding tipis di tengah layar dengan kecepatan yang tinggi. Di sana akan terlihat bahwa bola yang Anda gerakkan tidak menabrak dinding tersebut malahan menembus ke sisi sebelahnya. Hal ini terjadi karena tabrakan terjadi di tengah-tengah frame yang tidak dapat terdeteksi oleh program.
Perhatikan procedure GENERATE_PICTURE di sana sengaja dihasilkan bola yang kecil untuk mensimulasikan ketidaksempurnaan dari teknik ini. Demikian pula pada procedure MAIN, kecepatan yang digunakan jauh lebih besar dari biasanya sehingga memungkinkan bola berjalan dengan kecepatan yang sangat tinggi. Perlu diketahui bahwa sistem gerakan yang digunakan pada program ini tidak menggunakan sistem momentum untuk mensimulasikan situasi tabrakan di atas.

Walaupun bisa dikatakan teknik tabrakan antara dua benda dengan sistem perpotongan persegi panjang ini bekerja dengan baik untuk objek berukuran besar, dan berkecepatan lambat tetapi masih juga ada kekurangannya. Dengan teknik ini kita tidak peduli terhadap kejadian di tengah-tengah frame sehingga biasanya jika terdeteksi adanya tabrakan, kembalikan posisi objek ke posisi pada frame sebelumnya. Sedangkan jika kita bayangkan, posisi objek pada frame sebelumnya belum tentu tepat hampir menabrak objek yang kedua. Bisa saja masih ada jarak yang cukup jauh antara kedua objek ini sehingga animasi yang ditampilkan akan sangat jelek, karena bola berhenti tiba-tiba pada situasi tidak saling bersentuhan.
Untuk situasi ini coba perhatikan diagram di bawah ini.

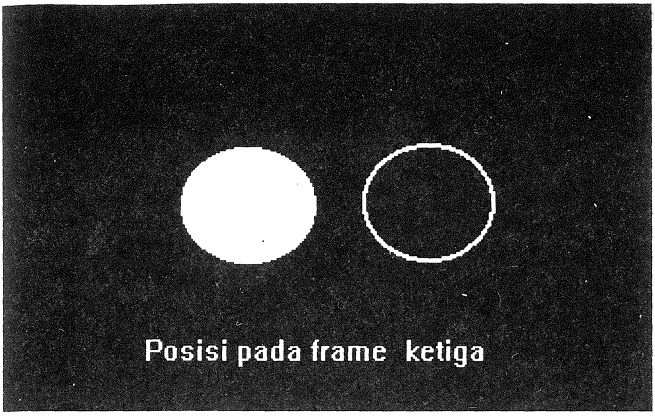
Pada frame pertama ini kedua bola sedang berjalan saling mendekati. Pada frame ini belum terjadi adanya tabrakan.



Kedua bola semakin mendekat dan belum terjadi tabrakan.



Kedua bola menabrak pada frame 3 sehingga posisi kedua bola dikembalikan ke posisi pada frame 2.



Sehingga pada frame 3 inilah posisi kedua bola. Hal ini akan terlihat sangat jelek karena pada layar monitor kita melihat dua bola tersebut belum bertabrakan walaupun sebenarnya sudah. Teknik ini masih digunakan pada game-game kuno tetapi sekarang hampir tidak ada yang menggunakan teknik ini.

Situasi di atas akan kita gambarkan pada program di bawah ini.

```
{-----------------------------------------------------}
{ Program: B_COL1.PAS                                 }
{ Coder : Agustinus Nalwan                            }
{ Fungsi: Mencoba sistem deteksi tabrakan antara      }
{         2 objek dengan sistem pixel                 }
{-----------------------------------------------------}

Uses Crt,Dos,VgaCad;

Type Block_Rec = Record
       X1,Y1,X2,Y2:Integer;
     End;

Var X,Y:Integer;
    Block:Block_Rec;
    Proc:Pointer;
    SpdX:Integer;
    Pic:Pointer;
    Finish:Boolean;
    OldInt9:Pointer;
    Flag,Flag2,Flag3,Flag4,Flag5,Flag6:Boolean;
    Ch:Char;
    Key:Byte;

procedure int9; interrupt; assembler;

label
```

```
  quit, release, tes2, tes3, tes5, testd, testa, testb, testc, ;
  tes4, tes6, teste;
asm
  sti
  in    al,60h
  mov   key,al
  test  al,80h
  jnz   release
  cmp   al,75
  jne   tes2
  mov   flag,1
  jmp   quit

tes2:
  cmp   al,77
  jne   tes3
  mov   flag2,1
  jmp   quit

tes3:
  cmp   al,72
  jne   tes4
  mov   flag3,1
  jmp   quit

tes4:
  cmp   al,80
  jne   tes5
  mov   flag4,1
  jmp   quit

tes5:
  cmp   al,28
  jne   tes6
  mov   flag5,1
  jmp   quit

tes6:
  cmp   al,57
  jne   quit
  mov   flag6,1
  jmp   quit

release:
  cmp   al,203
  jne   testa
  mov   flag,0
  jmp   quit

testa:
  cmp   al,205
  jne   testb
  mov   flag2,0
  jmp   quit
```

```
testb:
  cmp   al,200
  jne   testc
  mov   flag3,0
  jmp   quit

testc:
  cmp   al,208
  jne   testd
  mov   flag4,0
  jmp   quit

testd:
  cmp   al,156
  jne   teste
  mov   flag5,0
  jmp   quit

teste:
  cmp   al,185
  jne   quit
  mov   flag6,0

quit:
  mov   al,20h
  out   20h,al
end;

Procedure Initialize;
  Begin
    OpenVga256;
    Install_Palette('Palette.Pal');
    GetMem(Proc,64000);
    GetMem(Pic,20000);
    Finish:=False;
    X:=160;Y:=150;
    SpdX:=0;
    GetIntVec($9,OldInt9);
    SetIntVec($9,@int9);
  End;

Function Check_Collision(X,Vx:Integer):Boolean;
Var Rx:Integer;
  Begin
    Rx:=X+Vx;
    Check_Collision:=False;
    With Block do
      Begin
        If (Rx+40>=X1) and (Rx<=X2) then Check_Collision:=True;
      End;
  End;

Procedure Main;
  Begin
```

```
    Repeat
      FillChar(Proc^,64000,0);
      PutImage(Proc^,X,Y,Pic^);
      VgaCad.Block(Proc^,Block.X1,Block.Y1,Block.X2,Block.Y2,15);
      MoveTo(Proc^,Screen^);
      If Flag then
        If Check_Collision(X,-15)=False then Dec(X,15);
      If Flag2 then
        If Check_Collision(X,15)=False then Inc(X,15);
      If Flag5 then Finish:=True;
      If X<0 then X:=0;
      If X>280 then X:=280;
      Delay(20);
    Until Finish;
  End;

Procedure Removal;
  Begin
    CloseVga256;
    SetIntVec($9,OldInt9);
  End;

Procedure Generate_Pic;
Var A:Integer;
  Begin
    FillChar(Screen^,64000,0);
    For A:=1 to 20 do
      Circle(Screen^,160,100,A,A+16);
    GetImage(Screen^,140,80,180,120,Pic^);
  End;

Procedure Generate_Data;
  Begin
    With Block do
      Begin
        X1:=20;
        Y1:=60;
        X2:=50;
        Y2:=180;
      End;
  End;

Begin
  Initialize;
  Generate_Pic;
  Generate_Data;
  Main;
  Removal;
End.
```

PENJELASAN PROGRAM

Cobalah untuk menjalankan program di atas. Sekarang objek yang digerakkan cukup besar, dan kecepatan yang digunakkan tidak begitu besar. Gerakkan bola ke arah kiri untuk berusaha menembus dinding, di sana akan terlihat bahwa bola akan diberhentikan oleh situasi yang masih belum bersentuhan dengan dinding. Ini terjadi karena lokasi bola dikembalikan ke frame sebelumnya bila diketahui terjadi tabrakan.

Nah untuk itu kita akan membahas sebuah sistem pendeteksian tabrakan yang dapat memecahkan masalah di atas.

TEKNIK TABRAKAN DENGAN PENDETEKSIAN VEKTOR

TEKNIK PERHITUNGAN VEKTOR 1 SUMBU

Dengan melihat dari judul subtopik ini tentunya kita akan melakukan perhitungan yang berkenaan dengan vektor. Di sini ilmu matematika kita perlukan. Dengan melihat kembali situasi di atas, tentunya kita harus melakukan pendeteksian lokasi sebuah objek di antara 2 buah frame juga. Untuk itu kita harus menelusuri lompatan dari objek di antara 2 buah frame tersebut secara per-pixel. Misalnya lompatan dari objek tersebut dari frame 2 ke frame 3 adalah 50 pixel maka kita harus melakukan lompatan tersebut 1 pixel demi 1 pixel sambil memeriksa apakah objek tersebut telah menabrak objek lain. Jika objek tersebut menabrak objek lain, berhentikan objek tersebut pada posisi sebelumnya. Dengan demikian tabrakan antara kedua objek yang bergerak dengan kecepatan yang sangat tinggi tidak akan terlewatkan di tengah -tengah frame.

Coba lihat program berikut.

```
{-------------------------------------------------}
{ Program: B_COL2.PAS                             }
{ Coder : Agustinus Nalwan                        }
{ Fungsi: Mencoba sistem deteksi tabrakan antar   }
{           2 objek dengan sistem vektor          }
{-------------------------------------------------}

Uses Crt,Dos,VgaCad;

Type Block_Rec = Record
       X1,Y1,X2,Y2:Integer;
     End;

Var X,Y:Integer;
    Block:Block_Rec;
    Proc:Pointer;
    SpdX:Integer;
    Pic:Pointer;
    Finish:Boolean;
    OldInt9:Pointer;
    Flag,Flag2,Flag3,Flag4,Flag5,Flag6:Boolean;
    Ch:Char;
    Key:Byte;

procedure int9; interrupt; assembler;

label
  quit, release, tes2, tes3, tes5, testd, testa, testb, testc, ;
  tes4, tes6, teste;
asm
  sti
  in    al,60h
  mov   key,al
  test  al,80h
  jnz   release
  cmp   al,75
  jne   tes2
  mov   flag,1
  jmp   quit

tes2:
  cmp   al,77
  jne   tes3
  mov   flag2,1
  jmp   quit

tes3:
  cmp   al,72
  jne   tes4
  mov   flag3,1
  jmp   quit
```

```
tes4:
  cmp   al,80
  jne   tes5
  mov   flag4,1
  jmp   quit

tes5:
  cmp   al,28
  jne   tes6
  mov   flag5,1
  jmp   quit

tes6:
  cmp   al,57
  jne   quit
  mov   flag6,1
  jmp   quit

release:
  cmp   al,203
  jne   testa
  mov   flag,0
  jmp   quit

testa:
  cmp   al,205
  jne   testb
  mov   flag2,0
  jmp   quit

testb:
  cmp   al,200
  jne   testc
  mov   flag3,0
  jmp   quit

testc:
  cmp   al,208
  jne   testd
  mov   flag4,0
  jmp   quit

testd:
  cmp   al,156
  jne   teste
  mov   flag5,0
  jmp   quit

teste:
  cmp   al,185
  jne   quit
  mov   flag6,0
```

```
quit:
  mov   al,20h
  out   20h,al
end;

Procedure Initialize;
  Begin
    OpenVga256;
    Install_Palette('Palette.Pal');
    GetMem(Proc,64000);
    GetMem(Pic,20000);
    Finish:=False;
    X:=160;Y:=150;
    SpdX:=0;
    GetIntVec($9,OldInt9);
    SetIntVec($9,@int9);
  End;

Function Check_Collision(X,Vx:Integer;Var NewVx:Integer):Boolean;
Var XOld,Rx,A:Integer;
  Begin
    Check_Collision:=False;
    NewVx:=Vx;
    XOld:=X;
    If Vx<0 then Rx:=-1 else Rx:=1;
    For A:=1 to Abs(Vx) do
      Begin
        With Block do
          Begin
            If (X+Rx+40>=X1) and (X+Rx<=X2) then
              Begin
                Check_Collision:=True;
                NewVx:=X-XOld;
                A:=Abs(Vx);
              End Else X:=X+Rx;
          End;
      End;
  End;

Procedure Main;
Var NSpdX:Integer;
  Begin
    Repeat
      FillChar(Proc^,64000,0);
      PutImage(Proc^,X,Y,Pic^);
      VgaCad.Block(Proc^,Block.X1,Block.Y1,Block.X2,Block.Y2,15);
      MoveTo(Proc^,Screen^);
      If Flag then
        If SpdX>-15 then Dec(SpdX);
      If Flag2 then
        If SpdX<15 then Inc(SpdX);
      If Check_Collision(X,SpdX,NSpdX)=True then SpdX:=0;
      Inc(X,NSpdX);
      If Not Flag And Not Flag2 then
```

```
        Begin
          If SpdX>0 then Dec(SpdX);
          If SpdX<0 then Inc(SpdX);
        End;
      If Flag5 then Finish:=True;
      If X<0 then X:=0;
      If X>280 then X:=280;
      Delay(20);
    Until Finish;
  End;

Procedure Removal;
  Begin
    CloseVga256;
    SetIntVec($9,OldInt9);
  End;

Procedure Generate_Pic;
Var A:Integer;
  Begin
    FillChar(Screen^,64000,0);
    For A:=1 to 20 do
      Circle(Screen^,160,100,A,A+16);
    GetImage(Screen^,140,80,180,120,Pic^);
  End;

Procedure Generate_Data;
  Begin
    With Block do
      Begin
        X1:=20;
        Y1:=60;
        X2:=50;
        Y2:=180;
      End;
  End;

Begin
  Initialize;
  Generate_Pic;
  Generate_Data;
  Main;
  Removal;
End.
```

PENJELASAN PROGRAM

Program di atas telah menggunakan sistem pendeteksian tabrakan dengan sistem vektor. Coba sekarang gerakkan bola ke arah kiri layar. Di sini akan terlihat bahwa bola berhenti tepat di mana ia bersentuhan dengan dinding. WOW!! Canggih kan?

Yuk kita telusuri MAIN program di atas.

```
Procedure Main;
Var NSpdX:Integer;
```

Variabel NSpdX ialah variabel yang akan kita gunakan untuk mendapatkan kecepatan horizontal yang baru jika terjadi tabrakan.

```
Begin
  Repeat
    FillChar(Proc^,64000,0);
    PutImage(Proc^,X,Y,Pic^);
    VgaCad.Block(Proc^,Block.X1,Block.Y1,Block.X2,Block.Y2,15);
```

Gambar dinding ke virtual screen.

```
    MoveTo(Proc^,Screen^);
    If Flag then
      If SpdX>-15 then Dec(SpdX);
    If Flag2 then
      If SpdX<15 then Inc(SpdX);
```

Lakukan perubahan kecepatan jika terjadi penekanan tombol kiri atau kanan.

```
    If Check_Collision(X,SpdX,NSpdX)=True then SpdX:=0;
```

Procedure Check_Collision ini bertujuan untuk mendeteksi apakah benda telah menabrak dinding. Jika tabrakan terjadi, variabel NSpdX akan dikirimkan kembali dengan harga jarak antara objek (pada frame sebelum terjadinya tabrakan) dengan dinding. NSpdX ini dapat kita asumsikan sebagain Velocity yang boleh dipakai oleh objek, karena dengan menambah posisi objek dengan variabel ini maka objek akan berhenti tepat bersentuhan dengan dinding.
Jika tabrakan tidak terjadi maka NSpdX ini akan berharga sama dengan SpdX.
Jika tabrakan terjadi di mana objek telah bersentuhan dengan dinding, maka secara otomatis NSpdX yang menunjukkan jarak antara objek dengan dinding akan berharga 0 sehingga nantinya lokasi objek akan ditambahkan dengan 0.

Dengan cara ini tabrakan akan dapat dihitung dengan akurat sekali.

```
    Inc(X,NSpdX);
```

Karena NSpdX ini sudah menggantikan fungsi SpdX maka tambahkan koordinat X dari objek dengan NSpdX.

```
    If Not Flag And Not Flag2 then
      Begin
        If SpdX>0 then Dec(SpdX);
        If SpdX<0 then Inc(SpdX);
      End;
```

Jika tidak dilakukan penekanan tombol kiri atau kanan, set kecepatan perlahan-lahan ke 0.

```
    If Flag5 then Finish:=True;
```

Deteksi tombol ENTER untuk keluar dari program.

```
    If X<0 then X:=0;
    If X>280 then X:=280;
    Delay(20);
  Until Finish;
End;
```

Sekarang kita akan membahas function CHECK_COLLISION yang memegang peranan dalam mengetahui kapan tabrakan terjadi dan jarak antara objek dan dinding pada frame sebelum terjadinya tabrakan.

```
Function Check_Collision(X,Vx:Integer;Var NewVx:Integer):Boolean;
Var XOld,Rx,A:Integer;
  Begin
    Check_Collision:=False;
```

Asumsikan tabrakan tidak terjadi.

```
    NewVx:=Vx;
```

Asumsikan NewVx sama dengan Vx (anggap tidak terjadi tabrakan).

```
    XOld:=X;
```

Simpan posisi X yang lama.

```
    If Vx<0 then Rx:=-1 else Rx:=1;
```

Karena kita akan menelusuri lokasi benda dengan kecepatan yang diberikan 1 pixel demi 1 pixel maka kita harus mengetahui apakah kecepatan tersebut negatif yang berarti kita harus menelusuri ke arah kiri atau ke arah kanan jika Vx = positive. Variabel Rx di sini kita gunakan sebagai counter untuk melakukan penelusuran. Dengan Rx negatif maka secara otomatis penelusuran akan berjalan ke arah kiri demikian pula sebaliknya.

```
    For A:=1 to Abs(Vx) do
      Begin
```

Set loop sebanyak kecepatan (V). Karena kita akan menelusuri 1 pixel demi 1 pixel sebanyak kecepatan yang diberikan.

```
With Block do
  Begin
    If (X+Rx+40>=X1) and (X+Rx<=X2) then
```

Jika pada penelusuran berikutnya (X + Rx) terjadi tabrakan maka .....

```
Begin
  Check_Collision:=True;
  NewVx:=X-XOld;
  A:=Abs(Vx);
```

Set CHECK_COLLISION = TRUE.
Set NewVx = Posisi X sekarang dikurangi posisi X yang lama sebelum dilakukannya penelusuran.
Dengan cara ini variabel NewVx akan berisi jarak antara objek sebelum dilakukan penelusuran dengan dinding.

Keluar dari loop.

```
End Else X:=X+Rx;
```

Jika tidak terjadi tabrakan, tambahkan koordinat X dengan Rx untuk melanjutkan penelusuran.

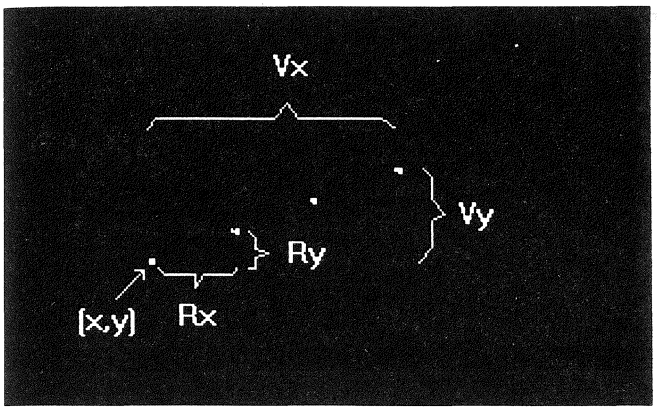
```
        End;
    End;
End;
```

Tidak terlalu sulit kan? Dengan dasar program di atas kita akan melanjutkan pembahasan kita pada teknik pendeteksian tabrakan 2 sumbu yaitu sumbu X dan sumbu Y mengingat pada program di atas kita masih menggunakan gerakan satu sumbu yaitu sumbu X.

TEKNIK PERHITUNGAN VEKTOR DENGAN 2 SUMBU

Nah, sekarang siapkan konsentrasi benar-benar karena teknik ini agak membingungkan, sampai-sampai saya sendiri bingung untuk menjelaskannya. Pada game-game bertipe SUPER MARIO BROSS, JACK JAZZ RABBIT dan kawan-kawannya, gerakan objek yang digunakan bukan hanya gerakan horizontal. Tetapi bisa vertikal maupun horizontal, malahan bisa juga gabungan keduanya. Dengan dasar perhitungan vektor satu sumbu, kita akan mennciptakan teknik perhitungan vektor pada dua sumbu. Anda sudah memiliki gambaran bagaimana caranya?

Ini berarti kita harus melakukan penelusuran secara pixel demi pixel ke arah vertikal maupun horizontal karena kali ini kecepatan yang akan kita telusuri ada dua yaitu VX dan VY. HORE!! penelusuran ini sama persis seperti teknik penggambaran garis yang telah kita bahas pada buku seri pertama. Untuk itu kita dapat menggunakan teknik BRESENHAM algorithm dalam melakukan penelusuran.



Perhatikan diagram di atas. Gambar tersebut menunjukkan situasi di mana kita melakukan penelusuran secara vertikal dan horizontal. Tiap titik di atas menunjukkan setiap loop dalam penelusuran. Rx dan Ry dihitung secara terpisah kemudian digunakan untuk menambah lokasi objek pada setiap loop.

Lebih baik kita mulai saja dengan program di bawah ini.

```
{-------------------------------------------------------}
{ Program: B_COL3.PAS                                   }
{ Coder  : Agustinus Nalwan                             }
{ Fungsi: Mencoba sistem tabrakan vektor dengan         }
{         tambahan analisa bagian objek yang tertabrak. }
{-------------------------------------------------------}

Uses Crt,Dos,VgaCad;

Type Block_Rec = Record
       X1,Y1,X2,Y2:Integer;
     End;

Var X,Y:Integer;
    Block:Block_Rec;
    Proc:Pointer;
    SpdX,SpdY:Integer;
    Pic:Pointer;
```

```
    Finish:Boolean;
    OldInt9:Pointer;
    Flag,Flag2,Flag3,Flag4,Flag5,Flag6:Boolean;
    Ch:Char;
    Key:Byte;

procedure int9; interrupt; assembler;

label
  quit, release, tes2, tes3, tes5, testd, testa, testb, testc,
  tes4, tes6, teste;
asm
  sti
  in    al,60h
  mov   key,al
  test  al,80h
  jnz   release
  cmp   al,75
  jne   tes2
  mov   flag,1
  jmp   quit

tes2:
  cmp   al,77
  jne   tes3
  mov   flag2,1
  jmp   quit

tes3:
  cmp   al,72
  jne   tes4
  mov   flag3,1
  jmp   quit

tes4:
  cmp   al,80
  jne   tes5
  mov   flag4,1
  jmp   quit

tes5:
  cmp   al,28
  jne   tes6
  mov   flag5,1
  jmp   quit

tes6:
  cmp   al,57
  jne   quit
  mov   flag6,1
  jmp   quit

release:
  cmp   al,203
```

```
  jne   testa
  mov   flag,0
  jmp   quit

testa:
  cmp   al,205
  jne   testb
  mov   flag2,0
  jmp   quit

testb:
  cmp   al,200
  jne   testc
  mov   flag3,0
  jmp   quit

testc:
  cmp   al,208
  jne   testd
  mov   flag4,0
  jmp   quit

testd:
  cmp   al,156
  jne   teste
  mov   flag5,0
  jmp   quit

teste:
  cmp   al,185
  jne   quit
  mov   flag6,0

quit:
  mov   al,20h
  out   20h,al
end;

Procedure Initialize;
  Begin
    OpenVga256;
    Install_Palette('Palette.Pal');
    GetMem(Proc,64000);
    GetMem(Pic,20000);
    Finish:=False;
    X:=160;Y:=150;
    SpdX:=0;SpdY:=0;
    GetIntVec($9,OldInt9);
    SetIntVec($9,@int9);
  End;

Function Collision(X1,Y1,XL,YL:Integer):Boolean;
  Begin
    If (X1+XL>Block.X1) and (X1<Block.X2) and
```

```
       (Y1+YL>Block.Y1) and (Y1<Block.Y2) then
         Collision:=True else Collision:=False;
  End;

Function Check_Move(Xr,Yr:Integer;
Var Vx,Vy,Rx,Ry:Integer;Xl,Yl:Integer):Byte;
Var A:Integer;
    Vrx,Vry:Real;
    Finish:Boolean;
    X,Y:Real;
    Loop,Count:Integer;
  Begin
    X:=Xr;Y:=Yr;
    If Abs(Vx)>Abs(Vy) then
      Begin
        If Vx>0 then Vrx:=1 else Vrx:=-1;
        Loop:=Abs(Vx);
        Vry:=Vy/Abs(Vx);
      End Else
    If Vy<>0 then
      Begin
        If Vy>0 then Vry:=1 else Vry:=-1;
        Vrx:=Vx/Abs(Vy);
        Loop:=Abs(Vy);
      End Else
      Begin
        Vry:=0;Vrx:=0;
        Loop:=1;
      End;
    Finish:=False;
    Count:=0;
    Check_Move:=0;
    Repeat
      Inc(Count);
      If Collision(Round(X+Vrx),Round(Vry+Y),Xl,Yl)=False then
        Begin
          X:=X+Vrx;
          Y:=Y+Vry;
        End Else
        Begin
          If Collision(Round(Vrx+X),Round(Y),Xl,Yl)=False then
            Begin
              { Height Collision }
              Finish:=True;
              If Vy>=0 then Check_Move:=2 else Check_Move:=4;
            End Else
          If Collision(Round(X),Round(Vry+Y),Xl,Yl)=False then
            Begin
              { Side Collision }
              Finish:=True;
              Check_Move:=1
            End Else
            Begin
              { Both Collision }
```

```
              Finish:=True;
              Check_Move:=3;
            End;
        End;
      If Count>=Loop then Finish:=True;
    Until Finish;
    Rx:=Round(X)-Xr;
    Ry:=Round(Y)-Yr;
  End;

Procedure Main;
Var NSpdX,NSpdY:Integer;
    Nol:Integer;
    Jump:Boolean;
    Tipe:Byte;
  Begin
    Jump:=False;
    Repeat
      FillChar(Proc^,64000,0);
      PutImage(Proc^,X,Y,Pic^);
      VgaCad.Block(Proc^,Block.X1,Block.Y1,Block.X2,Block.Y2,15);
      MoveTo(Proc^,Screen^);
      If Flag then
        If SpdX>-15 then Dec(SpdX);
      If Flag2 then
        If SpdX<15 then Inc(SpdX);
      If Flag3 and Not Jump then
        Begin
          SpdY:=-17;
          Jump:=True;
        End;
      Tipe:=Check_Move(X,Y,SpdX,SpdY,NSpdX,NSpdY,40,40);
      Inc(X,NSpdX);
      Inc(Y,NSpdY);
      Case Tipe of
        1:Begin          { Side }
            SpdX:=0;
          End;
        2:Begin          { Bottom }
            SpdY:=0;
            Jump:=False;
          End;
        4:Begin          { Top }
           SpdY:=-SpdY;
          End;
        3:Begin          { Both }
            SpdX:=0;
            SpdY:=0;
          End;
      End;
      If Y<10 then Y:=10;
      If Y>160 then
        Begin
          SpdY:=0;
```

```
          Y:=160;
          Jump:=False;
        End Else  Inc(SpdY);
      If Not Flag And Not Flag2 then
        Begin
          If SpdX>0 then Dec(SpdX);
          If SpdX<0 then Inc(SpdX);
        End;
      If Flag5 then Finish:=True;
      If X<0 then X:=0;
      If X>280 then X:=280;
      Delay(20);
    Until Finish;
  End;

Procedure Removal;
  Begin
    CloseVga256;
    SetIntVec($9,OldInt9);
  End;

Procedure Generate_Pic;
Var A:Word;
  Begin
    FillChar(Screen^,64000,0);
    Load_Sprite('PICS\DEMOSPR.PIC',Pic,A);
  End;

Procedure Generate_Data;
  Begin
    With Block do
      Begin
        X1:=60;
        Y1:=80;
        X2:=62;
        Y2:=120;
      End;
  End;

Begin
  Initialize;
  Generate_Pic;
  Generate_Data;
  Main;
  Removal;
End.
```

PENJELASAN PROGRAM

Program kita kali ini agak lain. Selain Anda dapat menggerakkan bola ke atas-bawah-kiri-kanan, gerakan bola secara horizontal menggunakan sistem momentum sedangkan gerakan vertical menggunakan sistem gravitasi. SISTEM GRAVITASI?? Teknik ini sih tidak begitu sulit untuk dipelajari. Teknik ini digunakan untuk mensimulasikan orang yang sedang melompat. Jika Anda perhatikan pada game-game yang beredar di pasaran atau paling gampang sih coba saja lompat di depan cermin. Perhatikan bahwa kecepatan Anda melompat mula-mula cepat tapi lambat laun semakin tinggi menjadi semakin lambat dan akhirnya Anda berhenti sejenak pada ketinggian maksimum dan akhirnya secara perlahan Anda mulai turun. Kecepatan turun ini makin lama makin cepat pula hingga Anda sudah mencapai permukaan tanah. Ini juga merupakan salah satu materi dari fisika yaitu Gerak Lurus Berubah Beraturan/GLBB. Yang menyebabkan hal ini terjadi ialah gravitasi bumi sekitar 10 m/s^2, kalau tidak salah ya. Pada saat kita melompat, dengan berdasarkan hukum kekekalan enerji (fisika nih ye), gaya yang kita lakukan dengan hentakan kaki akan berubah menjadi sebuah kecepatan awal. Katakanlah 20 m/s. Setiap detik, gravitasi bumi akan mengurangi kecepatan Anda sebanyak 10 m/s. Sehingga pada detik pertama kecepatan Anda tinggal 10 m/s^2, pada detik kedua tinggal 0 m/s. Pada saat ini Anda berada pada ketinggian maksimum. Dan pada detik ketiga, kecepatan Anda menjadi -10m/s. Pada saat ini Anda mulai jatuh ke bawah.
Karena gaya gravitasi hanya bekerja ke arah bawah maka kecepatan horizontal akan selalu konstan (tidak terpengaruh oleh gaya gravitasi).

Demikianlah sedikit gambaran mengenai Sistem Gravitasi.

Cobalah program di atas dan gerakkan bola ke kanan-kiri, dan tekan tombol atas untuk melompat. WOW!! Bagus sekali kan? Coba tabrakkan bola dengan dinding tipis di atas layar. Cobalah pula untuk menaikkan bola ke atas dinding tersebut. WOW!! Teknik inilah yang nantinya akan kita pergunakan untuk membuat game bertipe SUPER MARIO BROSS dan JACK JAZZ RABBIT.

Marilah kita mulai membahas program di atas.

```
Procedure Main;
Var NSpdX,NSpdY:Integer;
    Nol:Integer;
    Jump:Boolean;
    Tipe:Byte;
  Begin
    Jump:=False;
```

Variabel JUMP berfungsi untuk menyimpan kondisi dari bola. Jika bola sedang melompat maka set variabel JUMP = TRUE. Jika bola telah mencapai permukaan tanah maka set JUMP = FALSE. Dengan cara ini maka bola tidak akan dapat melompat pada saat sedang di udara. Kan lucu kalau pada saat lompat dan belum sampai kembali ke permukaan tanah, bola tersebut bisa lompat lagi.

```
Repeat
  FillChar(Proc^,64000,0);
  PutImage(Proc^,X,Y,Pic^);
  VgaCad.Block(Proc^,Block.X1,Block.Y1,Block.X2,Block.Y2,15);
  MoveTo(Proc^,Screen^);
```

Gambar dinding pada virtual screen.

```
If Flag then
  If SpdX>-15 then Dec(SpdX);
If Flag2 then
  If SpdX<15 then Inc(SpdX);
```

Lakukan pendeteksian tombol keyboard dan ubah kecepatan (sistem momentum).

```
If Flag3 and Not Jump then
  Begin
    SpdY:=-17;
    Jump:=True;
```

Jika tombol atas ditekan dan bola sedang dalam kondisi tidak melompat maka set SpdY yang merupakan kecepatan vertikal menjadi -17. Inilah kecepatan awal vertikal yang telah kita bahas di atas. Kecepatan ini akan semakin mendekati 0 karena terkena pengaruh gravitasi. Untuk mensimulasikan gravitasi, kita tinggal menambah SpdY ini pada setiap loop hingga SpdY ini menjadi 0.

```
  End;
Tipe:=Check_Move(X,Y,SpdX,SpdY,NSpdX,NSpdY,40,40);
```

Function CHECK_MOVE digunakan untuk mendeteksi tabrakan yang terjadi. Procedure ini agak lain dengan procedure yang kita gunakan untuk mendeteksi tabrakan pada program kita sebelumnya. Di sini diinputkan data-data dari sumbu X dan Y. Pada program kita terdahulu kita hanya memproses data pada sumbu X saja. Kedua parameter terakhir yaitu 40 dan 40 menunjukkan besar objek yang kita gerakkan dalam hal ini ialah bola.

Function ini akan menghasilkan angka:

- 0 : Jika tidak terjadi tabrakan.

- 1 : Jika terjadi tabrakan pada samping bola.
- 2 : Jika terjadi tabrakan pada bagian bawah bola.
- 3 : Jika terjadi tabrakan pada bagian samping dan bawah/atas bola.
- 4 : Jika terjadi tabrakan pada bagian atas bola.

Sebenarnya tujuan dari fungsi ini selain untuk mendapatkan kecepatan sumbu X dan kecepatan sumbu Y yang baru, juga untuk mendeteksi bagian mana dari bola yang tertabrak. Karena pada pembuatan game bertipe SUPER MARIO BROSS kita perlu mengetahui hal ini. Jika bagian atas/kepala si MARIO tertabrak balok maka MARIO akan terpantul ke arah bawah. Jika bagian bawah yang kena berarti MARIO telah mencapai permukaan suatu balok di mana dia dapat berdiri di sana. Sedangkan jika bagian samping yang kena, kecepatan horizontal si MARIO akan diset = 0, kecepatan vertikal akan tetap berjalan sebagaimana mestinya.

```
Inc(X,NSpdX);
Inc(Y,NSpdY);
```

Tambahkan koordinat X dan Y dengan speed yang telah dianalisa.

```
Case Tipe of
 1:Begin          { Side }
     SpdX:=0;
```

Jika bagian samping bola yang tertabrak maka 0-kan kecepatan horizontal.

```
   End;
 2:Begin          { Bottom }
     SpdY:=0;
     Jump:=False;
```

Jika bagian bawah bola yang tertabrak, berhentikan kecepatan vertikal. Set JUMP = FALSE karena dengan ini bola boleh melakukan lompatan lagi karena telah berhenti pada suatu permukaan.

```
   End;
 4:Begin          { Top }
    SpdY:=-SpdY;
```

Jika bagian atas bola yang tertabrak, ubah arah dari kecepatan vertikal karena kita ingin agar bola tersebut terpantul kembali ke bawah.

```
   End;
 3:Begin          { Both }
     SpdX:=0;
     SpdY:=0;
```

Jika terjadi tabrakan pada sisi samping dan vertikal maka set kedua kecepatan menjadi 0.

```
        End;
    End;
    If Y<10 then Y:=10;
```

Jika koordinat Y dari bola sudah mencapai atas layar maka berhentikan bola pada atas layar.

```
    If Y>160 then
      Begin
        SpdY:=0;
        Y:=160;
        Jump:=False;
```

Jika koordinat Y bola sudah berada di bawah layar, berhentikan bola dan set JUMP = FALSE karena dengan ini bola boleh melakukan lompatan lagi.

```
      End Else  Inc(SpdY);
```

Jika belum, simulasikan gravitasi yaitu dengan menambah kecepatan sumbu Y.

```
    If Not Flag And Not Flag2 then
      Begin
        If SpdX>0 then Dec(SpdX);
        If SpdX<0 then Inc(SpdX);
      End;
```

Jika tombol kiri atau kanan tidak ditekan, kerjakan sistem momentum pengereman pada kecepatan horizontal.

```
    If Flag5 then Finish:=True;
```

Keluar dari program jika ENTER ditekan.

```
    If X<0 then X:=0;
    If X>280 then X:=280;
    Delay(20);
  Until Finish;
End;
```

Sekarang kita akan menyelidiki cara kerja function CHECK_MOVE.

```
Function Check_Move(Xr,Yr:Integer;
Var Vx,Vy,Rx,Ry:Integer;Xl,Yl:Integer):Byte;
Var A:Integer;
    Vrx,Vry:Real;
    Finish:Boolean;
    X,Y:Real;
```

```
  Loop,Count:Integer;
Begin
  X:=Xr;Y:=Yr;
```

Simpan koordinat X dan Y ke dalam variabel REAL. Karena kita akan menggunakan bresenham algorithm di sini untuk menghitung gradient X dan Y. Sebenarnya kita dapat melakukan optimasi kecepatan pada procedure ini dengan menggunakan FIXED POINT ALGORITHM (ingat tidak?). Dengan menggeser satuan bilangan pecahan ke bilangan bulat yang lebih besar. Tetapi mengingat proses di bawah ini cukup rumit, dianjurkan untuk tidak melakukan optimasi lebih lanjut.

```
If Abs(Vx)>Abs(Vy) then
  Begin
    If Vx>0 then Vrx:=1 else Vrx:=-1;
    Loop:=Abs(Vx);
    Vry:=Vy/Abs(Vx);
  End Else
```

Jika magnitude dari Vx lebih besar dari Vy maka anggap Vx=1 atau -1 (jika arahnya ke kiri) dan Vy menjadi Vy dibagi magnitude dari Vx. Hal ini dilakukan agar kita dapat memperoleh lompatan yang tidak boleh lebih besar dari 1 pixel pada setiap loop penelusuran, karena jika ada 1 pixel saja yang terlewatkan, bisa-bisa teknik pendeteksian tabrakan kita akan gagal. Jumlah loop dari penelusuran ialah magnitude dari Vx.

```
If Vy<>0 then
  Begin
    If Vy>0 then Vry:=1 else Vry:=-1;
    Vrx:=Vx/Abs(Vy);
    Loop:=Abs(Vy);
```

Jika magnitude dari Vx lebih kecil, anggap Vy=1 atau -1 (jika arahnya ke atas) dan Vx menjadi Vx dibagi magnitude dari Vy. Jumlah loop penelusuran ialah magnitude dari Vy.

```
  End Else
  Begin
    Vry:=0;Vrx:=0;
    Loop:=1;
  End;
```

Jika terjadi Vy = 0 maka penelusuran akan berlangsung dengan lompatan 0 yang berarti tidak adanya lompatan. Hal ini bisa saja terjadi pada situasi di mana Vx dan Vy yang dikirimkan berharga 0.

```
Finish:=False;
Count:=0;
Check_Move:=0;
```

Anggap tidak terjadi tabrakan. Set Count = 0. Variabel Count ini akan berfungsi sebagai counter dari penelusuran.

```
Repeat
  Inc(Count);
```

Tambahkan counter penelusuran.

```
If Collision(Round(X+Vrx),Round(Vry+Y),Xl,Yl)=False then
  Begin
    X:=X+Vrx;
    Y:=Y+Vry;
```

Function COLLISION ini berguna untuk mengetahui apakah pada koordinat yang dikirimkan, objek tersebut menabrak dinding atau tidak.

Jika tidak terjadi tabrakan pada penelusuran berikutnya yaitu (X+Vrx, Y+Vry) maka tambahkan koordinat X dan Y dengan lompatan horizontal dan vertikal.

```
  End Else
  Begin
```

Jika terjadi tabrakan maka .....

```
If Collision(Round(Vrx+X),Round(Y),Xl,Yl)=False then
  Begin
    { Height Collision }
    Finish:=True;
    If Vy>=0 then Check_Move:=2 else Check_Move:=4;
```

Jika bukan tabrakan samping . Hal ini dapat kita buktikan dengan men-test koordinat (X+Vrx, Y) di mana belum dilakukannya lompatan pada koordinat Y. Karena tabrakan telah terjadi maka keluar dari loop penelusuran dan cek apakah tabrakan tersebut atas atau bawah. Tabrakan tersebut ialah tabrakan atas jika Vy berharga negatif, di mana arah gerak objek ke atas. Tabrakan bawah jika harga Vy >=0.

```
  End Else
If Collision(Round(X),Round(Vry+Y),Xl,Yl)=False then
  Begin
    { Side Collision }
    Finish:=True;
    Check_Move:=1
```

Jika test koordinat (X, Y+Vry) gagal maka tabrakan tersebut bukan tabrakan atas atau bawah, maka tabrakan yang terjadi ialah tabrakan samping. Kita tidak perlu mengetahui apakah itu tabrakan kanan atau kiri, sebenarnya bisa sih dengan mendeteksi kecepatan Vx tetapi untuk game bertipe ini hal itu tidaklah banyak berguna.

```
        End Else
        Begin
          { Both Collision }
          Finish:=True;
          Check_Move:=3;
```

Jika semua test di atas gagal maka tabrakan yang terjadi pastilah tabrakan pada dua sumbu.

```
          End;
      End;
  If Count>=Loop then Finish:=True;
```

Jika penelusuran telah selesai (koordinat (X+Vx, Y+Vy) telah dicapai) maka keluar dari loop penelusuran.

```
Until Finish;
Rx:=Round(X)-Xr;
Ry:=Round(Y)-Yr;
```

Hitung kecepatan Vx dan Vy yang baru yaitu dengan mengurangi posisi akhir dari penelusuran dengan posisi awal.

```
End;
```

Nah dengan ini, pembahasan teknik-teknik pemrograman kita telah berakhir. Sekaranglah saatnya untuk mempraktekkan semua teknik yang telah diajarkan dalam sebuah game. Ok, sampai jumpa pada bab berikutnya!!

BAB 6

ULASAN TENTANG VGACAD.TPU

Pada bab ini akan diberikan listing program dari VGACAD.PAS yang telah kita modifikasi beserta cara pemakaian dari setiap procedure-nya.

Unit VGACAD.TPU kali ini telah benar - benar handal untuk digunakan dalam pembuatan game - game yang setaraf dengan game - game yang dijual di pasaran.

```
{-------------------------------------------------}
{ Program : VGACAD.PAS                            }
{ Coder   : Agustinus Nalwan                      }
{ Fungsi  : Unit Animasi dan Game Profesional     }
{           Versi yang telah dimodifikasi         }
{           Versi 2.00                            }
{-------------------------------------------------}

{$G+}
Unit VGACAD;

INTERFACE

Uses Crt,Dos;

Const Screen      : Pointer = Ptr ($A000,0);

Type Palette_Cell = Record
       R , G , B : Byte;
     End;

     Palette_Type = Array[0..255] of Palette_Cell;
     Palette_Addr = ^Palette_Type;

     Scroll_Page  = Array[1..800] of Pointer;

     Parallax_Data = Record
       Page_X, Page_Y : Integer;
       Last_X, Last_Y : Integer;
       Length, Width  : Integer;
     End;

Var
     A,B             :  Integer;
     Reg             :  Registers;
     Total_Page      :  Byte;
     File_P          :  File;
     UserPalette     :  Palette_Type;
     DefaultPalette:  Palette_Type;
     File_Pal        :  File;
     User_Pal        :  Palette_Addr;
     Default_Pal     :  Palette_Addr;
     Active_Pal      :  Palette_Addr;
     Page_Scroll     :  Scroll_Page;
     Parallax_Dat    :  Parallax_Data;
     Page_Width      :  Integer;
```

```
    X,Y             : Word;
    Px2,Py2         : Word;
    Px3,Py3         : Word;
    Already         : Boolean;

{ Procedure Inisialisasi Graphic }

Procedure OpenVga256;
Procedure CloseVga256;

{ Procedure Manipulasi Layar }

Procedure FillWord      (Var PicBuff; Size:Word; Code:Byte);
Procedure MoveW         (Var Source; Var Dest; Size:Word );
Procedure Define_Page   (Var PicBuff : Pointer);
Procedure UnDefine_Page (Var PicBuff : Pointer);
Procedure Fill_Page     (Var PicBuff; Col : Byte);
Procedure MoveTo        (Var Source, Destination);

{ Procedure Konstruksi Bangun Geometri }

Procedure PutPixel   (Var PicBuff; X,Y : Integer; Color : Byte);
Function  GetPixel   (Var PicBuff; X,Y : Integer) : Byte;
Procedure Line       (Var PicBuff; X1,Y1,X2,Y2:integer; Col : Byte);
Procedure VLine      (Var PicBuff; X,Y,L:integer; Col : Byte);
Procedure Dot_Line   (Var   PicBuff;   X1,Y1,X2,Y2:integer;   Col   :
Byte;No:Integer;Tot_Dot:Integer);
Procedure Rectangle  (Var PicBuffer; X1,Y1,X2,Y2 : Integer;
                      Color : Byte);
Procedure Block      (Var PicBuff; X1,Y1,X2,Y2 : Integer;
                      Col : Byte);
Procedure Circle     (Var PicBuff; X,Y : Integer; R : Integer;
                      Col : Byte);
Procedure Ellipse    (Var PicBuff; X,Y : Integer; Rx,Ry : Integer;
                      Col : Byte);

{ Procedure Manipulasi Sprite }

Function  ImageSize      (X1,Y1,X2,Y2 : Integer) : Word;
Procedure GetImage       (Var PicBuff; X1,Y1,X2,Y2 : Integer;
                          Var Buffer);
Procedure Load_Sprite    (File_Name:String; Var Ptr:Pointer;
                          Var Size:Word);
Procedure Save_Sprite    (File_Name:String; Ptr:Pointer;Size:Word);
Procedure Load_Picture   (File_Name:String; Var Buffer);
Procedure Save_Picture   (File_Name:String; Var Buffer);
Procedure PutOld         (Var PicBuff;X,Y : Integer; Var Buffer);
Procedure PutImage       (Var PicBuff;X1,Y1:Integer;Var Buffer);
Procedure Put1Col        (Var PicBuff;X1,Y1:Integer;Col:Byte;
                          Var Buffer);
Procedure Put3D          (Var PicBuff;X,Y : Integer;XL,YL:Word;
                          Var Buffer);
Procedure T_PutImage     (Var PicBuff;X1,Y1:Integer;Var Buffer;
                          Var PalTable);
```

```
Procedure T_Put1Col       (Var PicBuff;X1,Y1:Integer;Col:Byte;
                           Var Buffer;Var PalTable);
Procedure T_Put3D         (Var PicBuff;X,Y : Integer;XL,YL:Word;
                           Var Buffer;Var PalTable);
Procedure WordBlockImage  (Var PicBuff;X,Y : Integer;Var Buffer);
Procedure ByteBlockImage  (Var PicBuff;X1,Y1:Integer;Var Buffer);

{ Procedure Transparansi }

Procedure Install_Transparent(PalTable:Pointer;Name:String);
Procedure Search_Transparent_Palette(Name:String);

{ Procedure Struktur Palette }

Procedure SetVisualPalette   (No,R,G,B:Byte);
Procedure GetVisualPalette   (No:Byte; Var R,G,B:Byte);
Procedure SetPalette     (Var Arr_Pal:Palette_Addr;No,Rr,Gg,Bb:Byte);
Procedure GetPalette         (Var Arr_Pal:Palette_Addr; No:Byte;
                              Var Rr,Gg,Bb:Byte);
Procedure Load_Palette       (Pal_Buff:Pointer; File_Name:String);
Procedure Save_Palette       (Pal_Buff:Pointer; File_Name:String);
Procedure Refresh_Palette;
Procedure Active_Palette     (Ac_Pal:Palette_Addr);
Procedure Restore_Palette;
Procedure Install_Palette    (Name:String);

{ Procedure Manipulasi Palette }

Procedure Set_All_Palette     (Pal_Buff:Palette_Addr; R,G,B : Byte);
Procedure Active_To_Palette   (Pal_Dest:Palette_Addr; Del:Byte);
Function  Metamorph_Palette   (Pal_Buff, Pal_Dest : Palette_Addr;
                               Rate:Byte) : Boolean;

{ Procedure Scroll Layar }

Procedure Load_Vert_Scroll    (Name:String);
Procedure Free_Vert_Scroll;
Procedure Put_Vert_Scroll     (Var PicBuff; Y:Integer);
Procedure Load_Horiz_Scroll   (Name:String);
Procedure Free_Horiz_Scroll;
Procedure Put_Horiz_Scroll    (Var PicBuff;Y:Integer);
Procedure Load_Parallax_Scroll (Name:String);
Procedure Put_Parallax_Scroll (X,Y:Integer;Var PicBuff;Var Temp);

{ Procedure Kompresi Sprite }

Procedure Color_Compress   (Var Source;Var Destination;
                            Var Size:Word);
Procedure Color_UnCompress (Var Source;Var Destination);
Procedure Zero_Compress    (Var Source;Var Destination;
                            Var Size:Word);
Procedure Zero_UnCompress  (Var Source;Var Destination);
Procedure Rll_Compress     (Var Source;Var Destination;
                            Var Size:Word);
```

```
Procedure RLL_UnCompress       (Var Source,Destination);

{ Procedure Pembantu }

Procedure GetAddr (Var PicBuff;Var Seg,Offs:Word);

Implementation

Procedure Hlines  (Var PicBuffer; X1,X2,Y : Integer; Color : Byte);
{ Horizontal line }
Begin
  If X1<0 then X1:=0;
  If X2<0 then X2:=0;
  If X1>319 then X1:=319;
  IF X2>319 then X2:=319;
  If Y<0 then Y:=0;
  If Y>199 then Y:=199;
asm
  Les di,PicBuffer
  Mov dx,Y
  Mov ax,dx
  Shl dx,1
  Shl dx,1
  Add dx,ax
  Shl dx,1
  Shl dx,1
  Shl dx,1
  Shl dx,1
  Shl dx,1
  Shl dx,1
  Add dx,X1
  Add di,dx
  Mov cx,X2
  Sub cx,X1
  Inc cx
  Jcxz @2
  Cld
  Mov al,Color
  Rep Stosb
  @2:
end; { HLine }
End;

Procedure VLines (Var PicBuffer; Y1,Y2,X : Integer; Color : Byte);
{ Vertical line }
Begin
  If Y1<0 then Y1:=0;
  If Y2<0 then Y2:=0;
  If Y1>199 then Y1:=199;
  If Y2>199 then Y2:=199;
  If X<0 then X:=0;
  If X>319 then X:=319;
asm
  Les di,PicBuffer
```

```
  Mov dx,Y1
  Mov ax,dx
  Shl dx,1
  Shl dx,1
  Add dx,ax
  Shl dx,1
  Shl dx,1
  Shl dx,1
  Shl dx,1
  Shl dx,1
  Shl dx,1
  Add dx,X
  Add di,dx
  Mov cx,Y2
  Sub cx,Y1
  Inc cx
  Jcxz @2
  Cld
  Mov al,Color
  @1:
    Stosb
    Add di,319
    Loop @1
  @2:
end; { VLine }
End;

Procedure Rectangle (Var PicBuffer; X1,Y1,X2,Y2 : Integer;
                     Color : Byte);
{ Rectangle }

begin
  if Y2=Y1 then HLines(PicBuffer,X1,X2,Y1,Color) else
  If X2=X1 then VLines(PicBuffer,Y1,Y2,X1,Color) else
  Begin
  If X2 - X1 = 1 then
  begin
    Vlines (PicBuffer,Y1,Y2,X1,Color);
            VLines (PicBuffer,Y1,Y2,X2,Color);
  end
  Else If Y2 - Y1 = 1 then
       begin
         HLines (PicBuffer,X1,X2,Y1,Color);
                 HLines (PicBuffer,X1,X2,Y2,Color);
       end
       Else begin
         HLines (PicBuffer,X1,X2,Y1,Color);
                 VLines (PicBuffer,Y1+1,Y2,X2,Color);
         HLines (PicBuffer,X1,X2-1,Y2,Color);
                 VLines (PicBuffer,Y1+1,Y2-1,X1,Color);
            end;
  End;
end; { Rect }
```

```
Function  ImageSize (X1,Y1,X2,Y2 : Integer) : Word;
  Begin
    ImageSize := (X2 - X1 + 1) * (Y2 - Y1 + 1) + 4;
  End;

Procedure GetImage (Var PicBuff; X1,Y1,X2,Y2 : Integer; Var Buffer);
ASSEMBLER;
  Asm
    Push ds           { Push Stack                      }
    Lds si,PicBuff    { Arahkan DS:SI ke PICBUFF        }
    Les di,Buffer     { Arahkan ES:DI ke BUFFER         }
    Mov dx,Y1         { Dx = Y awal                     }
    Mov ax,dx         { Ax = Dx                         }
    Shl dx,1
    Shl dx,1          { Dx = Dx * 4                     }
    Add dx,ax         { Dx = Dx + Ax                    }
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1          { Dx = Dx * 32                    }
    Add dx,X1         { Dx = Dx + X awal                }
    Add si,dx         { Offset = Offset + Dx            }
    Mov bx,X2         { Bx = X akhir                    }
    Sub bx,X1         { Bx = Bx - X awal                }
    Inc bx            { Bx = Bx + 1 = Panjang Gambar    }
    Mov [es:di],bx    { Tulis Panjang Gambar            }
    Inc di            { Offset = Offset + 1             }
    Inc di            { Offset = Offset + 1             }
    Mov cx,Y2         { Cx = Y akhir                    }
    Sub cx,Y1         { Cx = Cx - Y awal                }
    Inc cx            { Cx = Cx + 1 = Lebar Gambar      }
    Mov [es:di],cx    { Tulis Lebar Gambar              }
    Inc di            { Offset = Offset + 1             }
    Inc di            { Offset = Offset + 1             }
    Cld               { Counter Cx Mundur               }
    @1:
      Mov dx,cx       { Dx = Sisa lebar gambar          }
      Mov cx,bx       { Cx = Panjang gambar             }
      Mov ax,si       { Ax = Si                         }
      Rep Movsb       { Baca titik sejumlah Cx          }
      Mov si,ax       { Kembalikan Si ke Si awal        }
      Add si,320      { Si ke koordinat Y berikutnya    }
      Mov cx,dx       { Cx = Dx                         }
      Loop @1         { Cx-1, Ulang hingga Cx = 0       }
    Pop ds            { Pop Stack                       }
  End;

Procedure OpenVga256;
  Begin
    Reg.Ax:=$13;
    Intr($10,Reg);
  End;
```

```
Procedure MoveW(Var Source; Var Dest; Size:Word); ASSEMBLER;
  Asm
    Push ds           { Push Stack                      }
    Les di,Dest       { Arahkan ES:DI ke Dest           }
    Lds si,Source     { Arahkan DS:SI ke Source         }
    Mov Cx,Size       { Counter Cx sebesar Size         }
    Rep MovsW         { Pindahkan DS:SI ke ES:DI        }
    Pop ds            { Pop Stack                       }
  End;

Procedure FillWord(Var PicBuff; Size:Word; Code:Byte); ASSEMBLER;
  Asm
    Push ds           { Push Stack                      }
    Les di,PicBuff    { Arahkan ES:DI ke PicBuff        }
    Mov Cx,Size       { Counter sebesar Size            }
    Mov al,Code       { Al = Warna                      }
    Mov ah,Code       { Ah = Warna                      }
    Rep StosW         { Isi ES:DI dengan Warna          }
    Pop ds            { Pop Stack                       }
  End;

Procedure PutPixel (Var PicBuff; X,Y : Integer; Color : Byte);
  Begin
    If (X>=0) and (X<=319) and (Y>=0) and (Y<=199) then
    Begin
  Asm
    Push ds           { Push Stack                      }
    Les di,PicBuff    { Arahkan ES:DI ke PICBUFF        }
    Mov dx,Y          { Dx = Koordinat Y                }
    Mov ax,dx         { Ax = Dx                         }
    Shl dx,1          {                                 }
    Shl dx,1          { Dx = Dx * 4                     }
    Add dx,ax         { Dx = Dx + Ax                    }
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1          { Dx = Dx * 32                    }
    Add dx,X          { Dx = Dx + Koordinat X           }
    Add di,dx         { Offset = Offset  + Dx           }
    Mov al,Color      { Al = Warna titik                }
    Stosb             { Alokasi memori layar            }
    Pop ds            { Pop Stack                       }
    End;
   End;
  End;

Procedure VLine(Var PicBuff; X,Y,L : Integer; Col: Byte);
Begin
  If Y<0 then Y:=0;
  If Y>199 then Y:=199;
  If X<0 then X:=0;
  If X>319 then X:=319;
```

```
  If Y+L>199 then L:=199-Y;

  Asm
    Push ds           { Push Stack                          }
    Les di,PicBuff    { Arahkan ES:DI ke PICBUFF'           }
    Mov dx,Y          { Dx = Koordinat Y                    }
    Mov ax,dx         { Ax = Dx                             }
    Shl dx,1          {                                     }
    Shl dx,1          { Dx = Dx * 4                         }
    Add dx,ax         { Dx = Dx + Ax                        }
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1          { Dx = Dx * 32                        }
    Add dx,X          { Dx = Dx + Koordinat X               }
    Add di,dx         { Offset = Offset  + Dx               }
    mov cx,L
    mov al,col
    cld
    @1:
      Mov [es:di],al  { Al = Warna titik                    }
      add di,320
    loop @1
    Pop ds            { Pop Stack                           }
  End;
End;

Function  GetPixel (Var PicBuff; X,Y : Integer) : Byte; ASSEMBLER;
  Asm
    Push ds           { Push Stack                          }
    Les di,PicBuff    { Arahkan ES:DI ke PICBUFF            }
    Mov dx,Y          { Dx = Koordinat Y                    }
    Mov ax,dx         { Ax = Dx                             }
    Shl dx,1          {                                     }
    Shl dx,1          { Dx = Dx * 4                         }
    Add dx,ax         { Dx = Dx + Ax                        }
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1          { Dx = Dx * 32                        }
    Add dx,X          { Dx = Dx + Koordinat X               }
    Add di,dx         { Offset = Offset  + Dx               }
    Mov al,[es:di]    { Pindahkan warna titik ke function   }
    Pop ds            { Pop Stack                           }
  End;

Procedure Dot_Line (Var PicBuff; X1,Y1,X2,Y2:integer;
                    Col : Byte;No:Integer;Tot_Dot:Integer);
Var Dx,Dy,Loop,Ctr              : Integer;
    XPos,YPos                   : Real;
```

```
    Rx,Ry                          : Real;
  Begin
    Dx:=X2-X1;
    Dy:=Y2-Y1;
    Rx:=Dx/Tot_Dot;
    Ry:=Dy/Tot_Dot;
    For Ctr:=0 to Tot_Dot do
      Begin
        If Ctr<=No then
        Put-
Pixel(PicBuff,Round(Rx*Ctr)+X1,Round(Ry*Ctr)+Y1,Round(Random(255)))
else
        PutPixel(PicBuff,Round(Rx*Ctr)+X1,Round(Ry*Ctr)+Y1,Col);
        If Ctr=No then If Round(Random(4))=0 then
          Rectangle(PicBuff,Round(Rx*Ctr)+X1-1,Round(Ry*Ctr)+Y1-1,
          Round(Rx*Ctr)+X1+1,Round(Ry*Ctr)+Y1+1,Round(Random(255)));
      End;
  End;

Procedure Line(Var PicBuff; X1,Y1,X2,Y2:integer; Col : Byte);
Var Dx,Dy,Loop,Ctr             : Integer;
    XPos,YPos                  : Real;
    Rx,Ry                      : Real;
  Begin
    Dx:=X2-X1;
    Dy:=Y2-Y1;
    XPos:=X1;YPos:=Y1;
      If Abs(Dx)>Abs(Dy) then
        Begin
          Rx:=1;
          If Dx<>0 then
            Begin
              Ry:=Abs(Dy/Dx);
              Loop:=Abs(Dx);
            End Else
            Begin
              Rx:=0;
              Ry:=1;
              Loop:=Abs(Dy);
            End;
        End Else
        Begin
          Ry:=1;
          If Dy<>0 then
            Begin
              Rx:=Abs(Dx/Dy);
              Loop:=Abs(Dy);
            End Else
            Begin
              Ry:=0;
              Rx:=1;
              Loop:=Abs(Dx)
            End;
        End;
```

```
    If Dx<0 then Rx:=-Rx;
    If Dy<0 then Ry:=-Ry;
    If (Round(XPos)>=0) and (Round(XPos)<=319) and
    (Round(YPos)>=0) and (Round(YPos)<=199) then
      PutPixel(PicBuff,Round(XPos),Round(YPos),Col);
    For Ctr:=1 to Loop do
      Begin
        XPos:=Xpos+Rx;
        YPos:=YPos+Ry;
    If (Round(XPos)>=0) and (Round(XPos)<=319) and
    (Round(YPos)>=0) and (Round(YPos)<=199) then
      PutPixel(PicBuff,Round(XPos),Round(YPos),Col);
      End;
 End;

Procedure Block(Var PicBuff; X1,Y1,X2,Y2 : Integer; Col : Byte);
Begin
  If X1<0 then X1:=0;
  If X1>319 then X1:=319;
  If X2<0 then X2:=0;
  If X2>319 then X2:=319;
  If Y1<0 then Y1:=0;
  If Y1>199 then Y1:=199;
  If Y2<0 then Y2:=0;
  If Y2>199 then Y2:=199;
  Asm
    Push ds           { Push Stack                      }
    Les di,PicBuff    { Arahkan DS:SI ke PICBUFF        }
    Mov dx,Y1         { Dx = Y awal                     }
    Mov ax,dx         { Ax = Dx                         }
    Shl dx,1
    Shl dx,1          { Dx = Dx * 4                     }
    Add dx,ax         { Dx = Dx + Ax                    }
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1          { Dx = Dx * 32                    }
    Add dx,X1         { Dx = Dx + X awal                }
    Add di,dx         { Offset = Offset + Dx            }
    Mov bx,X2         { Bx = X akhir                    }
    Sub bx,X1         { Bx = Bx - X awal                }
    Inc bx            { Bx = Bx + 1 = Panjang Blok      }
    Mov cx,Y2         { Cx = Y akhir                    }
    Sub cx,Y1         { Cx = Cx - Y awal                }
    Inc cx            { Cx = Cx + 1 = Lebar Blok        }
    Mov Al,Col        { Al = Warna                      }
    Cld               { Counter Cx Mundur               }
    @1:
      Mov dx,cx       { Dx = Sisa lebar Blok            }
      Mov cx,bx       { Cx = Panjang Blok               }
      Mov si,di       { Ax = Si                         }
      Rep Stosb       { Baca titik sejumlah Cx          }
```

```
      Mov di,si       { Kembalikan Si ke Si awal       }
      Add di,320      { Si ke koordinat Y berikutnya   }
      Mov cx,dx       { Cx = Dx                        }
      Loop @1         { Cx-1, Ulang hingga Cx = 0      }
    Pop ds            { Pop Stack                      }
  End;
End;

Procedure Circle (Var PicBuff; X,Y : Integer; R : Integer;
                  Col : Byte);
Var Teta    : Integer;
  Begin
    For Teta:=0 to 1440 do
      PutPixel(PicBuff,X+Round(R*Cos(Teta*pi/720)),;
               Y+Round(R*Sin(Teta*pi/720)),Col)
  End;

Procedure Ellipse (Var PicBuff; X,Y : Integer; Rx,Ry : Integer;
                   Col : Byte);
Var Teta    : Integer;
  Begin
    For Teta:=0 to 1440 do
      PutPixel(PicBuff,X+Round(Rx*Cos(Teta*pi/720)),;
               Y+Round(Ry*Sin(Teta*pi/720)),Col)
  End;

Procedure Load_Sprite(File_Name:String;Var Ptr:Pointer;
                      Var Size:Word);
  Begin
    Assign(File_P,File_Name);
    Reset(File_P,1);
    GetMem(Ptr,FileSize(File_P));
    BlockRead(File_P,Ptr^,FileSize(File_P));
    Size:=FileSize(File_P);
    Close(File_P);
  End;

Procedure Save_Sprite(File_Name:String;Ptr:Pointer;Size:Word);
  Begin
    Assign(File_P,File_Name);
    Rewrite(File_P,1);
    BlockWrite(File_P,Ptr^,Size);
    Close(File_P);
  End;

Procedure Define_Page(Var PicBuff : Pointer);
  Begin
    GetMem(PicBuff,64000);
  End;

Procedure UnDefine_Page(Var PicBuff : Pointer);
  Begin
    FreeMem(PicBuff,64000);
  End;
```

```
Procedure Fill_Page (Var PicBuff; Col : Byte);
  Begin
    FillWord(PicBuff,32000,Col);
  End;

Procedure MoveTo (Var Source, Destination);
  Begin
    MoveW(Source,Destination,32000);
  End;

Procedure Load_Picture (File_Name:String;Var Buffer);
  Begin
    Assign(File_P,File_Name);
    Reset(File_P,1);
    BlockRead(File_P,Buffer,64000);
    Close(File_P);
  End;

Procedure Save_Picture (File_Name:String;Var Buffer);
  Begin
    Assign(File_P,File_Name);
    Rewrite(File_P,1);
    BlockWrite(File_P,Buffer,64000);
    Close(File_P);
  End;

Procedure SetVisualPalette (No,R,G,B:Byte);
  Begin
    Port[$3c8]:=No;       { Port nomor Palette }
    Port[$3c9]:=R;        { Isi Port Merah     }
    Port[$3c9]:=G;        { Isi Port Hijau     }
    Port[$3c9]:=B;        { Isi Port Biru      }
  End;

Procedure GetVisualPalette (No:Byte; Var R,G,B:Byte);
  Begin
    Port[$3c8]:=No;       { Port nomor Palette }
    R:=Port[$3c9];        { Baca Port Merah    }
    G:=Port[$3c9];        { Baca Port Hijau    }
    B:=Port[$3c9];        { Baca Port Biru     }
  End;

Procedure SetPalette     (Var Arr_Pal:Palette_Addr; No,Rr,Gg,;
                           Bb:Byte);
  Begin
    With Arr_Pal^[No] do
      Begin
        R:=Rr;
        G:=Gg;
        B:=Bb;
      End;
  End;
```

```
Procedure GetPalette     (Var Arr_Pal:Palette_Addr; No:Byte;
                          Var Rr,Gg,Bb:Byte);
  Begin
    With Arr_Pal^[No] do
      Begin
        Rr:=R;
        Gg:=G;
        Bb:=B;
      End;
  End;

Procedure Load_Palette(Pal_Buff:Pointer; File_Name:String);
Var X         : Byte;
    R,G,B     : Char;
    RR,GG,BB  : Byte;
  Begin
    Assign(File_Pal,File_Name);
    Reset(File_Pal,1);
    BlockRead(File_Pal,Pal_Buff^,768);
    Close(File_Pal);
  End;

Procedure Save_Palette (Pal_Buff:Pointer; File_Name:String);
Var X         : Byte;
    Fil       : Text;
  Begin
    Assign(File_Pal,File_Name);
    Rewrite(File_Pal,1);
    BlockWrite(File_Pal,Pal_Buff^,768);
    Close(File_Pal);
  End;

Procedure Refresh_Palette;
  Begin
    For A:=0 to 255 do
      With Active_Pal^[A] do
        Begin
          SetVisualPalette(A,R,G,B);
        End;
  End;

Procedure Active_Palette(Ac_Pal:Palette_Addr);
  Begin
    MoveW(Ac_Pal^,Active_Pal^,384);
    Refresh_Palette;
  End;

Procedure Restore_Palette;
  Begin
    MoveW(Default_Pal^,Active_Pal^,384);
  End;

Procedure Install_Palette(Name:String);
  Begin
```

```
    GetMem(Default_Pal,768);
    GetMem(User_Pal,768);
    GetMem(Active_Pal,768);
    Load_Palette(Default_Pal,Name);
    Active_Palette(Default_Pal);
  End;

Procedure Set_All_Palette   (Pal_Buff:Palette_Addr; R,G,B : Byte);
  Begin
    For A:=0 to 255 do
      SetPalette(Pal_Buff,A,R,G,B);
  End;

Procedure Active_To_Palette(Pal_Dest:Palette_Addr; Del:Byte);
Var Done     : Boolean;
    A        : Integer;
  Begin
    Repeat
      Done:=True;
      Delay(Del);
      For A:=0 to 255 do
        With Pal_Dest^[A] do
          Begin
            If Active_Pal^[A].R<R then Inc(Active_Pal^[A].R);
            If Active_Pal^[A].R>R then Dec(Active_Pal^[A].R);
            If Active_Pal^[A].G<G then Inc(Active_Pal^[A].G);
            If Active_Pal^[A].G>G then Dec(Active_Pal^[A].G);
            If Active_Pal^[A].B<B then Inc(Active_Pal^[A].B);
            If Active_Pal^[A].B>B then Dec(Active_Pal^[A].B);
            If Active_Pal^[A].R<>R then Done:=False;
            If Active_Pal^[A].G<>G then Done:=False;
            If Active_Pal^[A].B<>B then Done:=False;
          End;
      Refresh_Palette;
    Until Done;
  End;

Function Metamorph_Palette(Pal_Buff, Pal_Dest : ;
                           Palette_Addr; Rate:Byte) : Boolean;
  Begin
    Metamorph_Palette:=True;
      For A:=0 to 255 do
        With Pal_Dest^[A] do
          Begin
            If Pal_Buff^[A].R<R then
              Begin
                If Pal_Buff^[A].R+Rate<R then;
                   Inc(Pal_Buff^[A].R,Rate)
                  Else Pal_Buff^[A].R:=R;
              End;
            If Pal_Buff^[A].R>R then
              Begin
                If Pal_Buff^[A].R-Rate>R then;
                   Dec(Pal_Buff^[A].R,Rate)
```

```
                Else Pal_Buff^[A].R:=R;
            End;
          If Pal_Buff^[A].G<G then
            Begin
              If Pal_Buff^[A].G+Rate<G then;
                 Inc(Pal_Buff^[A].G,Rate)
                Else Pal_Buff^[A].G:=G;
            End;
          If Pal_Buff^[A].G>G then
            Begin
              If Pal_Buff^[A].G-Rate>G then;
                 Dec(Pal_Buff^[A].G,Rate)
                Else Pal_Buff^[A].G:=G;
            End;
          If Pal_Buff^[A].B<B then
            Begin
              If Pal_Buff^[A].B+Rate<B then;
                 Inc(Pal_Buff^[A].B,Rate)
                Else Pal_Buff^[A].B:=B;
            End;
          If Pal_Buff^[A].B>B then
            Begin
              If Pal_Buff^[A].B-Rate>B then;
                 Dec(Pal_Buff^[A].B,Rate)
                Else Pal_Buff^[A].B:=B;
            End;
          If Pal_Buff^[A].R<>R then Metamorph_Palette:=False;
          If Pal_Buff^[A].G<>G then Metamorph_Palette:=False;
          If Pal_Buff^[A].B<>B then Metamorph_Palette:=False;
        End;
  End;

Procedure Load_Vert_Scroll (Name:String);
Var Size : Longint;
    Ctr  : Longint;
  Begin
    Ctr:=1;
    For A:=1 to 800 do
      GetMem(Page_Scroll[A],320);
    Assign(File_P,Name);
    Reset(File_P,1);
    Size:=FileSize(File_P);
    Page_Width:=Size div 320;
    While Size>64000 do
      Begin
        BlockRead(File_P,Page_Scroll[Ctr]^,64000);
        Size:=Size-64000;
        Inc(Ctr,200);
      End;
    BlockRead(File_P,Page_Scroll[Ctr]^,Size);
    Close(File_P);
  End;

Procedure Free_Vert_Scroll;
```

```
Var Size : Word;
  Begin
    For A:=1 to 800 do
      FreeMem(Page_Scroll[A],320);
  End;

Procedure Put_Part (Var Page_Scroll; Var PicBuff;
                    Y,Width : Integer); ASSEMBLER;
  Asm
    Push ds             { Push Stack                        }
    Les di,PicBuff      { Arahkan ES:DI ke PICBUFF          }
    Lds si,Page_Scroll  { Arahkan DS:SI ke Page_Scroll      }
    Mov dx,Y            { Dx = Y awal                       }
    Mov ax,dx           { Ax = Dx                           }
    Shl dx,1
    Shl dx,1            { Dx = Dx * 4                       }
    Add dx,ax           { Dx = Dx + Ax                      }
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1            { Dx = Dx * 32                      }
    Add di,dx           { Offset = Offset + Dx              }

    Mov dx,Width        { Dx = Lebar sisa layar             }
    Mov ax,dx           { Ax = Dx                           }
    Shl dx,1
    Shl dx,1            { Dx = Dx * 4                       }
    Add dx,ax           { Dx = Dx + Ax                      }
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1            { Dx = Dx * 16                      }
    Mov Cx,Dx           { Counter Cx sejumlah Dx            }
    Rep MovsW           { Pindahkan DS:SI ke ES:DI          }
    Pop ds              { Pop Stack                         }
  End;

Procedure Put_Vert_Scroll (Var PicBuff; Y:Integer);
  Begin
    If (Y<=Page_Width) and (Y>=1) then
      Begin
        If Page_Width-Y>=200 then
          MoveW(Page_Scroll[Y]^,PicBuff,32000)
        Else Begin
          MoveW(Page_Scroll[Y]^,PicBuff,(Page_Width-Y+1)*160);
          If Page_Width-Y+i<>200 then
            Put_Part(Page_Scroll[1]^,PicBuff,Page_Width-Y+1,;
                     200-(Page_Width-Y+1));
        End;
      End;
  End;
```

```
Procedure Load_Horiz_Scroll (Name:String);
Var Size:Word;
    Fil :File;
  Begin
    Assign(Fil,Name);
    Reset(Fil,1);
    BlockRead(Fil,X,2);
    BlockRead(Fil,Y,2);
    GetMem(Page_Scroll[1],64000);
    BlockRead(Fil,Page_Scroll[1]^,64000);
    Px2:=0;Py2:=0;
    Px3:=0;Py3:=0;
    {$I-}
      BlockRead(Fil,X,2);
      BlockRead(Fil,Y,2);
    {$I+}
      If (IOResult=0) and (X>0) then
        Begin
          Px2:=X;Py2:=Y+1;
          Size:=(pY2)*(pX2);
          GetMem(Page_Scroll[2],Size);
          BlockRead(Fil,Page_Scroll[2]^,Size);
        End;
    Close(Fil);
  End;

Procedure Free_Horiz_Scroll;
Var Size : Word;
  Begin
    FreeMem(Page_Scroll[1],64000);
    Size:=(Py2)*(Px2);
    If Px2<>0 then
      FreeMem(Page_Scroll[2],Size);
  End;

Procedure Put_XPart(Var Picbuff,Proc;X,Start,CtrX,XGrid:Word);
                    Assembler;
  Asm
    Push ds
    Les di,PicBuff
    Lds si,Proc
    Mov Cx,200
    @1:
      Push si
      Push di
      Push cx
      Add di,x
      Add si,Start
      Mov Cx,CtrX
      Rep MovsB
      Pop cx
      Pop di
      Pop si
      Add si,XGrid
```

```
      Add di,320
    Loop @1
    Pop ds
  End;

Procedure Put_Horiz_Scroll (Var PicBuff;Y:Integer);
Var S,L,T:Word;
  Begin
    If Y<320 then
      Put_XPart(PicBuff,Page_Scroll[1]^,0,Y,320-Y,320);
    If (Y<640) and (Y>0) and (Px2<>0) then
      Begin
        If Y>=320 then
          Begin
            S:=Y-320;
            L:=640-Y;
            T:=0;
          End Else
          Begin
            S:=0;
            L:=Y;
            T:=320-Y;
          End;
        Put_XPart(PicBuff,Page_Scroll[2]^,T,S,L,Px2);
      End;
  End;

Procedure Load_Parallax_Scroll (Name:String);
Var No_Pic:Integer;
    File_P:File;
    File_D:Text;
    Size  :Word;
    PCTR  :Integer;
  Begin
    Assign(File_P,Name+'.Scr');
    Reset(File_P,1);
    Assign(File_D,Name+'.Atr');
    Reset(File_D);
    With Parallax_Dat do
      Begin
        Readln(File_D,Length);
        Readln(File_D,Width);
        Readln(File_D,Page_X);
        Readln(File_D,Page_Y);
        Readln(File_D,Last_X);
        Readln(File_D,Last_Y);
      End;
    PCTR:=1;
    Repeat
    Readln(File_D,Size);
    GetMem(Page_Scroll[PCTR],Size);
    BlockRead(File_P,Page_Scroll[PCTR]^,Size);
    Inc(PCTR);
    Until Eof(File_D);
```

```
    Close(File_P);
    Close(File_D);
  End;

Procedure Put_Parallax_Scroll(X,Y:Integer;Var PicBuff;Var Temp);
Var Al_X,Al_Y : Integer;
    Xs,Ys     : Integer;
    Xr,Yr     : Integer;
    Pic_No    : Integer;
    Size      : Word;
  Begin
    Al_X:=(X Div 80)+1;
    Al_Y:=(Y Div 50)+1;
    Xs:=(-(X Mod 80));
    Ys:=(-(Y Mod 50));
    Xr:=Xs;Yr:=Ys;
    Repeat
      Repeat
        Pic_No:=((Al_Y-1)*Parallax_Dat.Page_X)+Al_X;
        Rll_UnCompress(Page_Scroll[Pic_No]^,Temp);
        If (Xr<0) or (Xr>239) or (Yr<0) or (Yr>150)
        then ByteBlockImage(PicBuff,xr,yr,Temp) Else
          WordBlockImage(PicBuff,xr,yr,Temp);
        Inc(Xr,80);
        Inc(Al_X);
      Until Xr>319;
      Xr:=Xs;
      Al_X:=(X Div 80)+1;
      Inc(Al_Y);
      Inc(Yr,50);
    Until Yr>199;
  End;


Procedure Install_Transparent(PalTable:Pointer;Name:String);
Var FileTable:File;
  Begin
    Assign(FileTable,Name);
    Reset(FileTable,1);
    BlockRead(FileTable,PalTable^,65280);
    Close(FileTable);
  End;

Procedure PutOld (Var PicBuff;X,Y : Integer; Var Buffer); ASSEMBLER;
  Asm
    Push ds            { Push Stack                }
    Les di,PicBuff     { Arahkan ES:DI ke PICBUFF  }
    Lds si,Buffer      { Arahkan DS:SI ke BUFFER   }
    Mov bx,[ds:si]     { Baca Panjang              }
    Inc si             { Offset + 1                }
    Inc si             { Offset + 1                }
    Mov cx,[ds:si]     { Baca Lebar                }
    Inc si             { Offset + 1                }
    Inc si             { Offset + 1                }
```

```
    Mov dx,Y            { Masukkan Y ke Dx          }
    Mov ax,dx           { Masukkan Dx ke Ax         }
    Shl dx,1
    Shl dx,1            { Dx = DX * 4               }
    Add dx,ax           { Dx = Dx + Ax              }
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1            { Dx = Dx * 32                    }
    Add dx,X            { Dx = Dx + X                     }
    Add di,dx           { Start Offset peletakkan titik   }
    Cld                 { Counter Cx Mundur               }
    @1:
      Mov dx,cx         { Dx = Sisa Lebar Gambar          }
      Mov cx,bx         { Cx = Sisa Panjang Gambar        }
      Push di           { Push Stack                      }
      @2:
        Lodsb           { Baca titik dari BUFFER          }
        Or al,al        { Apakah titik = 0                }
        Jz @3           { Kalau Y tidak perlu diletakkan  }
        Stosb           { Letakkan titik                  }
        Dec di          { Kurangi offset pembacaan titik karena di }
                          bawah akan ditambah lagi        }
        @3:
          Inc di        { Tambah offset pembacaan titik   }
        Loop @2         { CX-1, Ulang sampai sisa panjang gambar }
                        { CX = 0                          }
      Pop di            { Pop Stack                       }
      Add di,320        { Register Di ke koordinat Y berikutnya }
      Mov cx,dx         { Isi CX dengan sisa lebar gambar }
      Loop @1           { CX-1, Ulang hingga sisa lebar   }
                        { gambar  = 0                     }
    Pop ds              { Pop Stack                       }
  End;

Procedure PutImage(Var PicBuff;X1,Y1:Integer;Var Buffer);Assembler;
Var XWStart:Word;
    XRStart:Word;
    YWStart:Word;
    YRStart:Word;
    Length :Word;
    Width  :Word;
    XRead  :Word;  { CX }
    YRead  :Word;
  Asm
    Push ds
    Lds si,Buffer
    Les di,PicBuff
    Mov Ax,[ds:si]
    Mov Length,Ax
    Mov XRead,Ax
    Add si,2
```

```
Mov Ax,[ds:si]
Mov Width,Ax
Mov YRead,Ax
Mov YRStart,0
Mov Ax,Y1
Mov YWStart,Ax
Add si,2
Mov XRStart,0
Mov Ax,X1
Mov XWStart,Ax
Cmp Ax,0
Jge @X1_Not_Negative
  Neg Ax
  Mov XRStart,Ax
  Mov XWStart,0
  Sub XRead,Ax
@X1_Not_Negative:
  Mov Ax,X1
  Add Ax,Length
  Dec Ax
  Cmp Ax,319
Jle @X1_No_Right_Crop
  Sub Ax,319
  Sub XRead,Ax
@X1_No_Right_Crop:
  Mov Ax,Y1
  Cmp Ax,0
Jge @No_Top_Crop
  Neg Ax
  Mov YWStart,0
  Mov YRStart,Ax
  Sub YRead,Ax
@No_Top_Crop:
  Mov Ax,Y1
  Add Ax,Width
  Dec Ax
  Cmp Ax,199
Jle @No_Bot_Crop
  Sub Ax,199
  Sub YRead,Ax
@No_Bot_Crop:
  Add di,XWStart
  Mov Ax,YWStart
  Shl Ax,2
  Add Ax,YWStart
  Shl Ax,6
  Add di,Ax

  Mov Ax,YRStart
  Xor Ah,Ah
  Mul Length
  Add si,Ax
  Add si,XRStart
```

```
      Mov Cx,YRead
    @Loop1:
      Push Cx
      Push di
      Push si
      Mov Cx,XRead
      @Loop2:
         Lodsb
         or al,al
         Jz @No_Write
         Stosb
         dec di
       @No_Write:
         inc di
      Loop @Loop2
      Pop si
      Pop di
      Add si,Length
      Add di,320
      Pop Cx
      Loop @Loop1
    Pop ds
  End;

Procedure ByteBlockImage(Var PicBuff;X1,Y1:Integer;
                         Var Buffer);Assembler;
Var XWStart:Word;
    XRStart:Word;
    YWStart:Word;
    YRStart:Word;
    Length :Word;
    Width  :Word;
    XRead  :Word;  { CX }
    YRead  :Word;
  Asm
    Push ds
    Lds si,Buffer
    Les di,PicBuff
    Mov Ax,[ds:si]
    Mov Length,Ax
    Mov XRead,Ax
    Add si,2
    Mov Ax,[ds:si]
    Mov Width,Ax
    Mov YRead,Ax
    Mov YRStart,0
    Mov Ax,Y1
    Mov YWStart,Ax
    Add si,2
    Mov XRStart,0
    Mov Ax,X1
    Mov XWStart,Ax
    Cmp Ax,0
    Jge @X1_Not_Negative
```

```
  Neg Ax
  Mov XRStart,Ax
  Mov XWStart,0
  Sub XRead,Ax
@X1_Not_Negative:
  Mov Ax,X1
  Add Ax,Length
  Dec Ax
  Cmp Ax,319
Jle @X1_No_Right_Crop
  Sub Ax,319
  Sub XRead,Ax
@X1_No_Right_Crop:
  Mov Ax,Y1
  Cmp Ax,0
Jge @No_Top_Crop
  Neg Ax
  Mov YWStart,0
  Mov YRStart,Ax
  Sub YRead,Ax
@No_Top_Crop:
  Mov Ax,Y1
  Add Ax,Width
  Dec Ax
  Cmp Ax,199
Jle @No_Bot_Crop
  Sub Ax,199
  Sub YRead,Ax
@No_Bot_Crop:
  Add di,XWStart
  Mov Ax,YWStart
  Shl Ax,2
  Add Ax,YWStart
  Shl Ax,6
  Add di,Ax

  Mov Ax,YRStart
  Xor Ah,Ah
  Mul Length
  Add si,Ax
  Add si,XRStart

  Mov Cx,YRead
@Loop1:
  Push Cx
  Push di
  Push si
  Mov Cx,XRead
  Rep Movsb
  Pop si
  Pop di
  Add si,Length
  Add di,320
  Pop Cx
```

```
      Loop @Loop1
    Pop ds
  End;

Procedure Put1Col(Var PicBuff;X1,Y1:Integer;Col:Byte;
                  Var Buffer);Assembler;
Var XWStart:Word;
    XRStart:Word;
    YWStart:Word;
    YRStart:Word;
    Length :Word;
    Width  :Word;
    XRead  :Word;  { CX }
    YRead  :Word;
  Asm
    Push ds
    Lds si,Buffer
    Les di,PicBuff
    Mov Ax,[ds:si]
    Mov Length,Ax
    Mov XRead,Ax
    Add si,2
    Mov Ax,[ds:si]
    Mov Width,Ax
    Mov YRead,Ax
    Mov YRStart,0
    Mov Ax,Y1
    Mov YWStart,Ax
    Add si,2
    Mov XRStart,0
    Mov Ax,X1
    Mov XWStart,Ax
    Cmp Ax,0
    Jge @X1_Not_Negative
      Neg Ax
      Mov XRStart,Ax
      Mov XWStart,0
      Sub XRead,Ax
    @X1_Not_Negative:
      Mov Ax,X1
      Add Ax,Length
      Dec Ax
      Cmp Ax,319
    Jle @X1_No_Right_Crop
      Sub Ax,319
      Sub XRead,Ax
    @X1_No_Right_Crop:
      Mov Ax,Y1
      Cmp Ax,0
    Jge @No_Top_Crop
      Neg Ax
      Mov YWStart,0
      Mov YRStart,Ax
      Sub YRead,Ax
```

```
    @No_Top_Crop:
      Mov Ax,Y1
      Add Ax,Width
      Dec Ax
      Cmp Ax,199
    Jle @No_Bot_Crop
      Sub Ax,199
      Sub YRead,Ax
    @No_Bot_Crop:
      Add di,XWStart
      Mov Ax,YWStart
      Shl Ax,2
      Add Ax,YWStart
      Shl Ax,6
      Add di,Ax

      Mov Ax,YRStart
      Xor Ah,Ah
      Mul Length
      Add si,Ax
      Add si,XRStart

      Mov Cx,YRead
      Mov Bl,Col
    @Loop1:
      Push Cx
      Push di
      Push si
      Mov Cx,XRead
      @Loop2:
          Lodsb
          or al,al
          Jz @No_Write
          Mov [ES:DI],Bl
        @No_Write:
          inc di
      Loop @Loop2
      Pop si
      Pop di
      Add si,Length
      Add di,320
      Pop Cx
      Loop @Loop1
    Pop ds
  End;

Procedure T_PutImage(Var PicBuff;X1,Y1:Integer;Var Buffer;
                     Var PalTable);Assembler;
Var XWStart:Word;
    XRStart:Word;
    YWStart:Word;
    YRStart:Word;
    Length :Word;
    Width  :Word;
```

```
  XRead  :Word;  { CX }
  YRead  :Word;
Asm
  Push ds
  Lds si,Buffer
  Les di,PicBuff
  Mov Ax,[ds:si]
  Mov Length,Ax
  Mov XRead,Ax
  Add si,2
  Mov Ax,[ds:si]
  Mov Width,Ax
  Mov YRead,Ax
  Mov YRStart,0
  Mov Ax,Y1
  Mov YWStart,Ax
  Add si,2
  Mov XRStart,0
  Mov Ax,X1
  Mov XWStart,Ax
  Cmp Ax,0
  Jge @X1_Not_Negative
    Neg Ax
    Mov XRStart,Ax
    Mov XWStart,0
    Sub XRead,Ax
  @X1_Not_Negative:
    Mov Ax,X1
    Add Ax,Length
    Dec Ax
    Cmp Ax,319
  Jle @X1_No_Right_Crop
    Sub Ax,319
    Sub XRead,Ax
  @X1_No_Right_Crop:
    Mov Ax,Y1
    Cmp Ax,0
  Jge @No_Top_Crop
    Neg Ax
    Mov YWStart,0
    Mov YRStart,Ax
    Sub YRead,Ax
  @No_Top_Crop:
    Mov Ax,Y1
    Add Ax,Width
    Dec Ax
    Cmp Ax,199
  Jle @No_Bot_Crop
    Sub Ax,199
    Sub YRead,Ax
  @No_Bot_Crop:
    Add di,XWStart
    Mov Ax,YWStart
    Shl Ax,2
```

```
      Add Ax,YWStart
      Shl Ax,6
      Add di,Ax

      Mov Ax,YRStart
      Xor Ah,Ah
      Mul Length
      Add si,Ax
      Add si,XRStart

      Mov Cx,YRead
    @Loop1:
      Push Cx
      Push di
      Push si
      Mov Cx,XRead
      @Loop2:
          Lodsb
          or al,al
          Jz @No_Write

          Push bx
          Mov bl,[es:di]
          Xor bh,bh
          Xor ah,ah
          Dec al
          Shl ax,8
          Add ax,bx
          Push es
          Push di
          Les di,PalTable
          Add di,ax
          Mov al,[es:di]
          Pop di
          Pop es
          Pop bx

          Stosb
          dec di
        @No_Write:
          inc di
      Loop @Loop2
      Pop si
      Pop di
      Add si,Length
      Add di,320
      Pop Cx
      Loop @Loop1
    Pop ds
  End;

Procedure T_Put1Col(Var PicBuff;X1,Y1:Integer;Col:Byte;
                    Var Buffer;Var PalTable);Assembler;
Var XWStart:Word;
```

```
  XRStart:Word;
  YWStart:Word;
  YRStart:Word;
  Length :Word;
  Width  :Word;
  XRead  :Word;  { CX }
  YRead  :Word;
Asm
  Push ds
  Lds si,Buffer
  Les di,PicBuff
  Mov Ax,[ds:si]
  Mov Length,Ax
  Mov XRead,Ax
  Add si,2
  Mov Ax,[ds:si]
  Mov Width,Ax
  Mov YRead,Ax
  Mov YRStart,0
  Mov Ax,Y1
  Mov YWStart,Ax
  Add si,2
  Mov XRStart,0
  Mov Ax,X1
  Mov XWStart,Ax
  Cmp Ax,0
  Jge @X1_Not_Negative
    Neg Ax
    Mov XRStart,Ax
    Mov XWStart,0
    Sub XRead,Ax
  @X1_Not_Negative:
    Mov Ax,X1
    Add Ax,Length
    Dec Ax
    Cmp Ax,319
  Jle @X1_No_Right_Crop
    Sub Ax,319
    Sub XRead,Ax
  @X1_No_Right_Crop:
    Mov Ax,Y1
    Cmp Ax,0
  Jge @No_Top_Crop
    Neg Ax
    Mov YWStart,0
    Mov YRStart,Ax
    Sub YRead,Ax
  @No_Top_Crop:
    Mov Ax,Y1
    Add Ax,Width
    Dec Ax
    Cmp Ax,199
  Jle @No_Bot_Crop
    Sub Ax,199
```

```
  Sub YRead,Ax
@No_Bot_Crop:
  Add di,XWStart
  Mov Ax,YWStart
  Shl Ax,2
  Add Ax,YWStart
  Shl Ax,6
  Add di,Ax

  Mov Ax,YRStart
  Xor Ah,Ah
  Mul Length
  Add si,Ax
  Add si,XRStart

  Mov Cx,YRead
@Loop1:
  Push Cx
  Push di
  Push si
  Mov Cx,XRead
  @Loop2:
      Lodsb
      or al,al
      Jz @No_Write

      Push bx
      Mov bl,[es:di]
      Xor bh,bh
      Xor ah,ah
      Mov al,col
      Dec al
      Shl ax,8
      Add ax,bx
      Push es
      Push di
      Les di,PalTable
      Add di,ax
      Mov al,[es:di]
      Pop di
      Pop es
      Pop bx

      Stosb
      dec di
    @No_Write:
      inc di
  Loop @Loop2
  Pop si
  Pop di
  Add si,Length
  Add di,320
  Pop Cx
  Loop @Loop1
```

```
    Pop ds
  End;

Procedure T_Put3D(Var PicBuff;X,Y : Integer;XL,YL:Word;
                  Var Buffer;Var PalTable);
Var XWStart:Word;
    YWStart:Word;
    Length :Word;
    Width  :Word;
    XRead  :Word;  { CX }
    YRead  :Word;

    XFixed :Word;
    YFixed :Word;
    XFStart:Word;
    YFStart:Word;
    RX     :Word;
    RY     :Word;
  Begin
  Asm
    Push ds
    Lds si,Buffer
    Les di,PicBuff
    Mov Ax,[ds:si]
    Mov Length,Ax
    Mov Ah,Al
    Xor Al,Al
    Mov Bx,Ax
    Xor Dx,Dx
    Div XL
    Mov Rx,Ax
    Mov Ax,Bx
    Xor Dx,Dx
    Div Rx
    Mov XRead,Ax
    Add si,2
    Mov Ax,[ds:si]
    Mov Width,Ax
    Mov Ah,Al
    Xor Al,Al
    Mov Bx,Ax
    Xor Dx,Dx
    Div YL
    Mov Ry,Ax
    Mov Ax,Bx
    Xor Dx,Dx
    Div Ry
    Mov YRead,Ax
    Mov XFixed,0
    Mov YFixed,0
    Mov XFStart,0
    Mov YFStart,0
    Mov Ax,Y
    Mov YWStart,Ax
```

```
Add si,2
Mov Ax,X
Mov XWStart,Ax
Cmp Ax,0
Jge @X1_Not_Negative
  Neg Ax
  Mov XWStart,0
  Sub XRead,Ax
  Mul Rx
  Mov XFStart,Ax
@X1_Not_Negative:
  Mov Ax,XWStart
  Add Ax,XRead
  Dec Ax
  Cmp Ax,319
Jle @X1_No_Right_Crop
  Sub Ax,319
  Sub XRead,Ax
@X1_No_Right_Crop:
  Mov Ax,Y
  Cmp Ax,0
Jge @No_Top_Crop
  Neg Ax
  Mov YWStart,0
  Sub YRead,Ax
  Mul Ry
  Mov YFStart,Ax
@No_Top_Crop:
  Mov Ax,YWStart
  Add Ax,YRead
  Cmp Ax,200
Jle @No_Bot_Crop
  Sub Ax,200
  Sub YRead,Ax
@No_Bot_Crop:
  Add di,XWStart
  Mov Ax,YWStart
  Shl Ax,2
  Add Ax,YWStart
  Shl Ax,6
  Add di,Ax

  Mov Cx,YRead
  Mov Ax,YFStart
  Mov YFixed,Ax
@Loop1:
  Push Cx
  Push di
  Push si
  Mov Ax,YFixed
  Mov Al,Ah
  Xor Ah,Ah
  Mul Length
  Add si,Ax
```

```
      Mov Cx,XRead
      Mov Ax,XFStart
      Mov XFixed,Ax
      Mov Bx,Rx
      @Loop2:
         Mov Ax,XFixed
         Mov Al,Ah
         Xor Ah,Ah
         Push si
         Add si,Ax

         Lodsb
         Pop si
         Add XFixed,Bx
         or al,al
         Jz @No_Write
          { Transparent Algorithm }
          Push bx
          Mov bl,[es:di]
          Xor bh,bh
          Xor ah,ah
          Dec al
          Shl ax,8
          Add ax,bx
          Push es
          Push di
          Les di,PalTable
          Add di,ax
          Mov al,[es:di]
          Pop di
          Pop es
          Pop bx
          { ----------}
         Stosb
         dec di
       @No_Write:
         inc di
      Loop @Loop2
      Pop si
      Pop di
      Mov Bx,Ry
      Add YFixed,Bx
      Add di,320
      Pop Cx
      Loop @Loop1
    Pop ds
  End;
  End;

Procedure Put3D(Var PicBuff;X,Y : Integer;XL,YL:Word; Var Buffer);
Var XWStart:Word;
    YWStart:Word;
    Length :Word;
```

```
    Width   :Word;
    XRead   :Word;  { CX }
    YRead   :Word;

    XFixed :Word;
    YFixed :Word;
    XFStart:Word;
    YFStart:Word;
    RX      :Word;
    RY      :Word;
  Begin
  Asm
    Push ds
    Lds si,Buffer
    Les di,PicBuff
    Mov Ax,[ds:si]
    Mov Length,Ax
    Mov Ah,Al
    Xor Al,Al
    Mov Bx,Ax
    Xor Dx,Dx
    Div XL
    Mov Rx,Ax
    Mov Ax,Bx
    Xor Dx,Dx
    Div Rx
    Mov XRead,Ax
    Add si,2
    Mov Ax,[ds:si]
    Mov Width,Ax
    Mov Ah,Al
    Xor Al,Al
    Mov Bx,Ax
    Xor Dx,Dx
    Div YL
    Mov Ry,Ax
    Mov Ax,Bx
    Xor Dx,Dx
    Div Ry
    Mov YRead,Ax
    Mov XFixed,0
    Mov YFixed,0
    Mov XFStart,0
    Mov YFStart,0
    Mov Ax,Y
    Mov YWStart,Ax
    Add si,2
    Mov Ax,X
    Mov XWStart,Ax
    Cmp Ax,0
    Jge @X1_Not_Negative
      Neg Ax
      Mov XWStart,0
      Sub XRead,Ax
```

```
  Mul Rx
  Mov XFStart,Ax
@X1_Not_Negative:
  Mov Ax,XWStart
  Add Ax,XRead
  Dec Ax
  Cmp Ax,319
Jle @X1_No_Right_Crop
  Sub Ax,319
  Sub XRead,Ax
@X1_No_Right_Crop:
  Mov Ax,Y
  Cmp Ax,0
Jge @No_Top_Crop
  Neg Ax
  Mov YWStart,0
  Sub YRead,Ax
  Mul Ry
  Mov YFStart,Ax
@No_Top_Crop:
  Mov Ax,YWStart
  Add Ax,YRead
  Cmp Ax,200
Jle @No_Bot_Crop
  Sub Ax,200
  Sub YRead,Ax
@No_Bot_Crop:
  Add di,XWStart
  Mov Ax,YWStart
  Shl Ax,2
  Add Ax,YWStart
  Shl Ax,6
  Add di,Ax

  Mov Cx,YRead
  Mov Ax,YFStart
  Mov YFixed,Ax
@Loop1:
  Push Cx
  Push di
  Push si
  Mov Ax,YFixed
  Mov Al,Ah
  Xor Ah,Ah
  Mul Length
  Add si,Ax

  Mov Cx,XRead
  Mov Ax,XFStart
  Mov XFixed,Ax
  Mov Bx,Rx
  @Loop2:
     Mov Ax,XFixed
     Mov Al,Ah
```

```
        Xor Ah,Ah
        Push si
        Add si,Ax

        Lodsb
        Pop si
        Add XFixed,Bx
        or al,al
        Jz @No_Write
        Stosb
        dec di
      @No_Write:
        inc di
      Loop @Loop2
      Pop si
      Pop di
      Mov Bx,Ry
      Add YFixed,Bx
      Add di,320
      Pop Cx
      Loop @Loop1
    Pop ds
  End;
  End;

Procedure WordBlockImage  (Var PicBuff;X,Y : Integer;
                           Var Buffer);Assembler;
  Asm
    Push ds            { Push Stack                }
    Les di,PicBuff     { Arahkan ES:DI ke PICBUFF  }
    Lds si,Buffer      { Arahkan DS:SI ke BUFFER   }
    Mov bx,[ds:si]     { Baca Panjang              }
    Inc si             { Offset + 1                }
    Inc si             { Offset + 1                }
    Mov cx,[ds:si]     { Baca Lebar                }
    Inc si             { Offset + 1                }
    Inc si             { Offset + 1                }
    Mov dx,Y           { Masukkan Y ke Dx          }
    Mov ax,dx          { Masukkan Dx ke Ax         }
    Shl dx,1
    Shl dx,1           { Dx = DX * 4               }
    Add dx,ax          { Dx = Dx + Ax              }
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1
    Shl dx,1           { Dx = Dx * 32              }
    Add dx,X           { Dx = Dx + X               }
    Add di,dx          { Start Offset peletakkan titik }
    Shr bx,1
    Cld                { Counter Cx Mundur         }
    @1:
      Mov dx,cx        { Dx = Sisa Lebar Gambar    }
```

```
      Mov cx,bx          { Cx = Sisa Panjang Gambar }
      Push di            { Push Stack               }
      Rep MovsW          { Baca titik dari BUFFER   }
      Pop di             { Pop Stack                }
      Add di,320         { Register Di ke koordinat Y berikutnya }
      Mov cx,dx          { Isi CX dengan sisa lebar gambar }
      Loop @1            { CX-1, Ulang hingga sisa lebar gambar = 0}
    Pop ds               { Pop Stack                }
  End;

Procedure Search_Transparent_Palette(Name:String);
Var A,B,Col1,
    Col2           : Integer;
    Red1,Red2      : Integer;
    Green1,Green2  : Integer;
    Blue1,Blue2    : Integer;
    SRed,SGreen,
    SBlue          : Integer;

    DRed       : Integer;
    DGreen     : Integer;
    DBlue      : Integer;
    Red,Green,
    Blue       : Integer;
    DTotal     : Integer;
    DMin       : Integer;
    TransCol   : Byte;
    FileTable  : File;
    Ch         : Char;
  Begin
    Assign(FileTable,Name);
    Rewrite(FileTable,1);
    For Col1:=1 to 255 do
      Begin
        Red1  :=Default_Pal^[Col1].R;
        Green1:=Default_Pal^[Col1].G;
        Blue1 :=Default_Pal^[Col1].B;
        For Col2:=0 to 255 do
          Begin
            Red2  :=Default_Pal^[Col2].R;
            Green2:=Default_Pal^[Col2].G;
            Blue2 :=Default_Pal^[Col2].B;

            SRed  :=(Red1+Red2)     div 2;
            SGreen:=(Green1+Green2) div 2;
            SBlue :=(Blue1+Blue2)   div 2;
            DMin:=1000;
            For A:=0 to 255 do
              Begin
                Red  :=Default_Pal^[A].R;
                Green:=Default_Pal^[A].G;
                Blue :=Default_Pal^[A].B;
```

```
                DRed   :=Abs(Red-SRed);
                DGreen :=Abs(Green-SGreen);
                DBlue  :=Abs(Blue-SBlue);
                DTotal:=DRed+DGreen+DBlue;
                If (DTotal<DMin) or (DTotal = 0) then
                  Begin
                    DMin:=DTotal;
                    TransCol:=A;
                  End;
              End;
            BlockWrite(FileTable,TransCol,1);
          End;
        End;
      Close(FileTable);
    End;

Procedure GetAddr(Var PicBuff;Var Seg,Offs:Word);
Var Sg,Off:Word;
  Begin
    Asm
      push ds
      les di,PicBuff
      mov Sg,es
      mov Off,di
      pop ds
    End;
    Seg:=Sg+Off div 16;
    Offs:=Off Mod 16;
  End;

Procedure Color_Compress(Var Source;Var Destination;Var Size:Word);
Var SegS,OffS   :Word;
    SegD,OffD   :Word;
    Length,Width:Word;
    Total       :Word;
    Finish      :Boolean;
    Value       :Byte;
    Long        :Byte;
  Begin
    GetAddr(Source,SegS,OffS);
    GetAddr(Destination,SegD,OffD);
    Move(Mem[SegS:OffS],Length,2);
    Move(Mem[SegS:OffS+2],Width,2);
    Total:=Width*Length+4;
    Finish:=False;

    Value:=Mem[SegS:OffS];
    Long:=0;
    Inc(OffD,2);
    Size:=0;
    Repeat
      If (Mem[SegS:OffS]=Value) and (Long<255) and (OffS<Total) then
        Begin
          Inc(Long);
```

```
        Inc(OffS);
      End Else
      Begin
        Mem[SegD:OffD]:=Value;
        Inc(OffD);
        Mem[SegD:OffD]:=Long;
        Inc(OffD);
        Inc(Size,2);
        Value:=Mem[SegS:OffS];
        Long:=0;
        If OffS>=Total then Finish:=True;
      End;
  Until Finish;
  Move(Size,Destination,2);
  Inc(Size,2);
End;

Procedure Color_UnCompress(Var Source;Var Destination);Assembler;
  Asm
    Push ds
    Lds si,Source
    Les di,Destination
    LodsW
    Shr Ax,1
    Mov Cx,Ax
    @Loop1:
      Push Cx
      LodsW
      Xor Ch,Ch
      Mov Cl,Ah
      @Loop2:
        Stosb
      Loop @Loop2
      Pop Cx
    Loop @Loop1
    Pop ds
  End;

Procedure Zero_Compress(Var Source;Var Destination;Var Size:Word);
Var SegS,OffS    :Word;
    SegD,OffD    :Word;
    Length,Width:Word;
    Total        :Word;
    Finish       :Boolean;
    Value        :Byte;
    Long         :Byte;
    Packet       :Word;
    A,Ctr        :Word;
  Begin
    GetAddr(Source,SegS,OffS);
    GetAddr(Destination,SegD,OffD);
    Move(Mem[SegS:OffS],Length,2);
    Move(Mem[SegS:OffS+2],Width,2);
    Total:=Width*Length+4;
```

```
    Finish:=False;

    Value:=Mem[SegS:OffS];
    Long:=0;
    Inc(OffD,2);
    Size:=2;
    Packet:=0;
    Finish:=False;
    Ctr:=0;
    Repeat
      If Value=0 then
        Begin
          Long:=0;
          Repeat
            Inc(Long);
            Inc(OffS);
            Inc(Ctr);
          Until (Mem[SegS:OffS]<>0) or (Long>=255) or (Ctr>=Total);
          Mem[SegD:OffD]:=0;
          Mem[SegD:OffD+1]:=Long;
          Inc(OffD,2);
          Value:=Mem[SegS:OffS];
          Inc(Packet);
          Inc(Size,2);
        End else
        Begin
          Mem[SegD:OffD]:=Value;
          Inc(OffD);
          Inc(OffS);
          Value:=Mem[SegS:OffS];
          Inc(Packet);
          Inc(Ctr);
          Inc(Size);
        End;
      If Ctr>=Total then Finish:=True;
    Until Finish;
    GetAddr(Destination,SegD,OffD);
    Move(Packet,Mem[SegD:OffD],2);
  End;

Procedure Zero_UnCompress(Var Source;Var Destination);
  Begin
  Asm
    Push ds
    Lds si,Source
    Les di,Destination
    LodsW
    Mov Cx,Ax     { Get Packed }
    @Loop1:
      Push Cx
      Lodsb
      Mov Cx,1
      Or Al,Al
      Jnz @Loop2
```

```
      Mov bl,al
      Lodsb
      Mov cl,al
      Mov al,bl
      @Loop2:
        Stosb
      Loop @Loop2
      Pop Cx
    Loop @Loop1
    Pop ds
  End;
  End;

Procedure Rll_Compress(Var Source;Var Destination;Var Size:Word);
Var SegS,OffS:Word;
    SegD,OffD:Word;
    Finish:Boolean;
    SrcCtr:Byte;
    FByte :Byte;
    Ctr   :Word;
    OldOFF:Word;
    Length,Width:Word;
  Begin
    Ctr:=1;
    GetAddr(Source,SegS,OffS);
    GetAddr(Destination,SegD,OffD);
    Move(Mem[SegS:OffS],Length,2);
    Move(Mem[SegS:OffS+2],Width,2);
    Size:=Length*Width+4;
    OldOFF:=OffD;
    Finish:=False;
    FByte:=Mem[SegS:OffS];
    Repeat
      SrcCtr:=1;
      While (FByte=Mem[SegS:OffS+SrcCtr]) and (SrcCtr<255) and
      (Ctr<=Size) do
        Begin
          Inc(SrcCtr);
          Inc(Ctr);
        End;
      If SrcCtr>1 then
        Begin
          Mem[SegD:OffD]:=SrcCtr;
          Inc(OffD);
          Mem[SegD:OffD]:=FByte;
          Inc(OffD);
          Inc(OffS,SrcCtr);
          FByte:=Mem[SegS:OffS];
          Inc(Ctr);
        End Else
        Begin
          Mem[SegD:OffD]:=0;
          SrcCtr:=0;
          While (Mem[SegS:OffS+SrcCtr]<>Mem[SegS:OffS+SrcCtr+1]) and
```

```
          (SrcCtr<255) and (Ctr<=Size) do
            Begin
              Inc(Ctr);
              Mem[SegD:OffD+SrcCtr+2]:=Mem[SegS:OffS+SrcCtr];
              Inc(SrcCtr);
            End;
          Mem[SegD:OffD+1]:=SrcCtr;
          Inc(OffD,SrcCtr+2);
          Inc(OffS,SrcCtr);
          FByte:=Mem[SegS:OffS];
        End;
      If Ctr>Size then Finish:=True;
    Until Finish;
    Mem[SegD:OffD]:=0;
    Mem[SegD:OffD+1]:=0;
    Size:=OffD+2-OldOFF;
  End;

Procedure RLL_UnCompress (Var Source,Destination); ASSEMBLER;
asm
  Mov dx,ds              { save ds }
  Les di,Destination
  Lds si,Source
  Xor bx,bx
  Cld

@1:
  Xor cx,cx
  Lodsb
  Or  al,al
  Jz  @2
  Mov cl,al
  Add bx,cx
  Lodsb
  Rep Stosb
  Loop @1

@2:
  Lodsb
  Or  al,al
  Jz  @3
  Mov cl,al
  Add bx,cx
  Rep Movsb
  Loop @1

@3:
  Mov ds,dx
end; { RLLxUnCompress }

Procedure CloseVga256;
  Begin
    TextMode(80);
  End;
```

```
Begin
  Randomize;
  Already:=False;
End.
```

DAFTAR FUNGSI DAN PROCEDURE DARI VGACAD.PAS

Procedure Animasi Tiga Layar

Procedure : FillWord (Var PicBuff; Size:Word; Code:Byte);
Fungsi : Mengisi virtual screen dengan sebuah warna secara WORD.
Parameter : PicBuff = Alamat virtual screen yang dituju.
Size = Jumlah word yang akan diisikan.
Code = Warna yang akan anda pakai.

Procedure : MoveW (Var Source; Var Dest; Size:Word);
Fungsi : Memindahkan virtual screen ke virtual screen yang lain.
Parameter : Source = Alamat virtual screen asal.
Dest = Alamat virtual screen tujuan.
Size = Besar virtual screen yang dipakai.

Procedure : Define_Page (Var PicBuff : Pointer);
Fungsi : Menciptakan sebuah virtual screen dengan alamat PicBuff.
Parameter : PicBuff = Output yang menunjukkan alamat memori yang digunakan oleh virtual screen tersebut.

Procedure : UnDefine_Page (Var PicBuff : Pointer);
Fungsi : Melenyapkan sebuah virtual screen dengan alamat PicBuff.
Parameter : PicBuff = Alamat virtual screen yang akan dilenyapkan.

Procedure : Fill_Page (Var PicBuff; Col : Byte);
Fungsi : Untuk menghapus sebuah virtual screen dengan satu warna.
Parameter : PicBuff = Alamat virtual screen yang dituju.
Col = Warna yang akan diisikan.

Procedure : MoveTo (Var Source, Destination);
Fungsi : Memindahkan sebuah virtual screen ke virtual screen yang lain.
Parameter : Source = Alamat virtual screen asal.
Destination = Alamat virtual screen tujuan.

Procedure Konstruksi Bangun Geometri

Procedure : PutPixel (Var PicBuff; X,Y : Integer; Color : Byte);
Fungsi : Menciptakan titik pada virtual screen.
Parameter : PicBuff = Alamat virtual screen yang dituju.
X,Y = Koordinat titik.
Col = Warna titik.

Procedure : GetPixel (Var PicBuff; X,Y : Integer) : Byte;
Fungsi : Membaca sebuah titik pada virtual screen.
Parameter : PicBuff = Alamat virtual screen yang dituju.
X,Y = Koordinat titik.
Return : Warna titik.

Procedure : Line (Var PicBuff; X1,Y1,X2,Y2:integer; Col : Byte);
Fungsi : Menciptakan garis pada virtual screen.
Parameter : PicBuff = Alamat virtual screen yang dituju.
X1,Y1 = Koordinat start awal garis.
X2,Y2 = Koordinat akhir garis.
Col = Warna garis.

Procedure : Vline (Var PicBuff; X,Y,L:integer; Col : Byte);
Fungsi : Menciptakan garis vertikal pada virtual screen.
Parameter : X,Y = Koordinat awal garis.
L = Panjang vertikal garis.
Col = Warna garis.

Procedure : Dot_Line (Var PicBuff; X1,Y1,X2,Y2:integer; Col: Byte;
No:Integer;Tot_Dot:Integer);
Fungsi : Menciptakan garis putus - putus pada virtual screen.
Parameter : PicBuff = Alamat virtual screen.
X1,Y1,X2,Y2 = Start awal dan akhir garis.
Col = Warna garis.
No = Nomor titik yang akan dilingkari dengan kotak.
Tot_Dot = Jumlah titik yang akan ditampilkan sepanjang garis tersebut.

Procedure : Rectangle (Var PicBuffer; X1,Y1,X2,Y2 : Integer;
Color : Byte);
Fungsi : Menggambar persegi panjang pada virtual screen.
Parameter : Picbuffer = Alamat virtual screen.

X1,Y1,X2,Y2 = Koordinat kiri atas dan kanan bawah persegi panjang.
Col = Warna persegi panjang.

Procedure : Block (Var PicBuff; X1,Y1,X2,Y2 : Integer; Col : Byte);
Fungsi : Menampilkan balok berisi pada virtual screen.
PicBuff = Alamat virtual screen.
X1,Y1,X2,Y2 = Koordinat kiri atas dan kanan bawah balok.
Col = Warna balok.

Procedure : Circle (Var PicBuff; X,Y : Integer; R : Integer; Col : Byte);
Fungsi : Menggambar lingkaran pada virtual screen.
Parameter : PicBuff = Alamat virtual screen.
X,Y = Koordinat pusat lingkaran.
R = Jari-jari lingkaran.
Col = Warna lingkaran.

Procedure : Ellipse (Var PicBuff; X,Y : Integer; Rx,Ry : Integer; Col : Byte);
Fungsi : Menggambar elips pada virtual screen.
Parameter : PicBuff = Alamat virtual screen.
X,Y = Koordinat pusat elips.
Rx, Ry = Jari - jari sumbu X dan Y.
Col = Warna elips.

Procedure Manipulasi Sprite

Procedure : ImageSize (X1,Y1,X2,Y2 : Integer) : Word;
Fungsi : Menghitung jumlah memori yang dibutuhkan oleh sprite.
Parameter : X1,Y1,X2,Y2 = Ukuran sprite.
Return : Jumlah memori yang diperlukan.

Procedure : GetImage (Var PicBuff; X1,Y1,X2,Y2 : Integer; Var Buffer);
Fungsi : Mengambil sprite dari virtual screen ke memori.
Parameter : PicBuff = Alamat virtual screen.
X1,Y1,X2,Y2 = Koordinat kiri atas dan kanan bawah sprite yang akan diambil.
Buffer = Alamat memori tempat sprite disimpan.

Procedure : Load_Sprite (File_Name:String; Var Ptr:Pointer; Var Size:Word);
Fungsi : Membaca sprite file.
Parameter : File_Name = Nama file tempat sprite disimpan.

Ptr = Alamat memori tempat sprite akan disimpan.
Size = Output yang menunjukkan besar sprite yang dibaca dari file.

Procedure : Save_Sprite (File_Name:String;Ptr: Pointer;Size:Word);
Fungsi : Menyimpan sprite ke dalam file.
Parameter : File_Name = Nama file tempat sprite akan disimpan.
Ptr = Alamat memori tempat sprite disimpan.
Size = Besar dari sprite yang akan disimpan dalam file.

Procedure : Load_Picture (File_Name:String; Var Buffer);
Fungsi : Untuk menampilkan sebuah gambar dari file pada virtual screen.
Parameter : File_Name = Nama file tempat gambar disimpan.
Buffer = Alamat dari virtual screen.

Procedure : Save_Picture (File_Name:String; Var Buffer);
Fungsi : Menyimpan gambar pada virtual screen ke file.
Parameter : File_Name = Nama file tempat gambar akan disimpan.
Buffer = Alamat virtual screen di mana gambar berada.

Procedure : PutOld (Var PicBuff;X,Y : Integer; Var Buffer);
Fungsi : Menampilkan sprite dengan tidak mempedulikan batasan layar.
Parameter : PicBuff = Alamat virtual screen.
X, Y = Koordinat di mana kita akan menampilkan sprite.
Buffer = Alamat memori tempat sprite disimpan.

Procedure : PutImage (Var PicBuff;X1,Y1:Integer;Var Buffer);
Fungsi : Menampilkan sprite dengan mempedulikan batasan layar.
Parameter : PicBuff = Alamat virtual screen.
X1, Y1 = Koordinat di mana kita akan menampilkan sprite.
Buffer = Alamat memori tempat sprite disimpan.

Procedure : Put1Col (Var PicBuff;X1,Y1:Integer;Col:Byte;Var Buffer);
Fungsi : Menampilkan sprite dengan satu warna.
Parameter : PicBuff = Alamat virtual screen.

X1, Y1 = Koordinat tempat sprite akan ditampilkan,
Col = Warna sprite.
Buffer = Alamat sprite pada memori.

Procedure : Put3D (Var PicBuff;X,Y : Integer;XL,YL:Word;Var Buffer);
Fungsi : Menampilkan sprite yang dapat di-resize.
Parameter : PicBuff = Alamat virtual screen.
X, Y = Koordinat di mana sprite akan ditampilkan.
XL,YL = Ukuran baru dari sprite yang diminta.
Buffer = Alamat memori di mana sprite disimpan.

Procedure : T_PutImage (Var PicBuff;X1,Y1:Integer;Var Buffer;
Var PalTable);
Fungsi : Menampilkan sprite secara transparant.
Parameter : PicBuff = Alamat virtual screen.
X1, Y1 = Koordinat di mana sprite akan ditampilkan.
Buffer = Alamat memori di mana sprite disimpan.
PalTable = Alamat memori dari tabel transparant.

Procedure : T_Put1Col (Var PicBuff;X1,Y1:Integer;Col:Byte;
Var Buffer;Var PalTable);
Fungsi : Menampilkan sprite secara transparant dan satu warna.
Parameter : PicBuff = Alamat virtual screen.
X1, Y1 = Koordinat di mana sprite akan ditampilkan.
Col = Warna sprite.
Buffer = Alamat sprite.
PalTable = Alamat tabel transparant.

Procedure : T_Put3D (Var PicBuff;X,Y : Integer;XL,YL:Word;
Var Buffer;Var PalTable);
Fungsi : Menampilkan sprite dengan teknik resize dan transparant.
Parameter : PicBuff = Alamat virtual screen.
X, Y = Koordinat di mana sprite akan ditampilkan.
XL, YL = Ukuran baru dari sprite yang diminta.
Buffer = Alamat di mana sprite disimpan.
PalTable = Alamat tabel transparant.

Procedure : WordBlockImage (Var PicBuff;X,Y : Integer;Var Buffer);
Fungsi : PutImage dengan sistem WORD dengan tidak mempedulikan warna 0 dan pembatasan layar.
Parameter : PicBuff = Alamat virtual screen.
X, Y = Koordinat sprite yang akan ditampilkan.

Buffer = Alamat sprite.

Procedure : ByteBlockImage (Var PicBuff;X1,Y1:Integer;Var Buffer);
Fungsi : PutImage dengan sistem BYTE dengan tidak mempedulikan warna 0.
Parameter : PicBuff = Alamat virtual screen.
X1, Y1 = Koordinat sprite yang akan ditampilkan.
Buffer = Alamat sprite.

Procedure Transparansi

Procedure : Install_Transparent (PalTable:Pointer;Name:String);
Fungsi : Membaca sebuah tabel transparant file.
Parameter : PalTable = Alamat tabel transparant.
Name = Nama file tabel transparant.

Procedure : Search_Transparent_Palette (Name:String);
Fungsi : Menciptakan sebuah tabel transparant dan menyimpannya ke dalam sebuah file.
Parameter : Name = Nama file yang akan digunakan untuk menyimpan tabel transparant.

Procedure Struktur Palette

Procedure : SetVisualPalette (No,R,G,B:Byte);
Fungsi : Mengubah susunan RGB dari sebuah palette yang aktif.
Parameter : No = Nomor Palette yang akan diubah.
R,G,B = Intensitas Red, Green dan Blue yang baru.

Procedure : GetVisualPalette (No:Byte; Var R,G,B:Byte);
Fungsi : Membaca susunan RGB dari sebuah palette yang aktif.
Parameter : No = Nomor Palette yang akan diambil.
R,G,B = Output Red, Green dan Blue yang terbaca.

Procedure : SetPalette (Var Arr_Pal:Palette_Addr;No,Rr,Gg,Bb:Byte);
Fungsi : Mengubah susunan RGB dari sebuah palette pada virtual palette.
Parameter : Arr_Pal = Alamat virtual palette.
No = Nomor palette yang akan diubah.
Rr,Gg,Bb = Intensitas Red, Green dan Blue yang baru.

Procedure : GetPalette (Var Arr_Pal:Palette_Addr; No:Byte; Var Rr,Gg,; Bb:Byte);
Fungsi : Membaca susunan RGB dari sebuah palette pada virtual palette.
Parameter : Arr_Pal = Alamat dari virtual palette.
No = Nomor palette yang akan dibaca.
Rr,Gg,Bb = Output dari Red, Green, Blue yang terbaca.

Procedure : Load_Palette (Pal_Buff:Pointer;File_Name:String);
Fungsi : Membaca susunan palette dari file.
Parameter : Pal_Buff = Alamat virtual palette yang akan digunakan untuk menyimpan susunan palette.
File_Name = Nama file tempat susunan palette disimpan.

Procedure : Save_Palette (Pal_Buff:Pointer; File_Name:String);
Fungsi : Menyimpan virtual palette ke dalam file.
Parameter : Pal_Buff = Alamat virtual palette yang akan disimpan.
File_Name = Nama file tempat virtual palette akan disimpan.

Procedure : Refresh_Palette;
Fungsi : Mengaktifkan palette pada virtual palette Active_Pal ke layar.

Procedure : Active_Palette (Ac_Pal:Palette_Addr);
Fungsi : Memindahkan isi virtual palette ke dalam Active_Pal.
Parameter : Ac_Pal = Alamat virtual palette yang akan dipindahkan ke Active_Pal.

Procedure : Restore_Palette;
Fungsi : Memindahkan isi Default_Pal ke dalam Active_Pal dan sekaligus mengaktifkan Active_Pal ke layar.

Procedure : Install_Palette (Name:String);
Fungsi : Membaca virtual palette dari file dan memasukkannya ke dalam Default_Pal.
Parameter : Name = Nama dari file yang akan dibaca.

Procedure Manipulasi Palette

Procedure : Set_All_Palette (Pal_Buff:Palette_Addr; R,G,B : Byte);
Fungsi : Mengisi semua palette pada virtual palette dengan RGB yang diberikan.
Parameter : Pal_Buff = Alamat virtual palette.

R, G, B = Intensitas yang akan diisikan.

Procedure : Active_To_Palette (Pal_Dest:Palette_Addr; Del:Byte);
Fungsi : Mengubah intensitas Active_Pal secara perlahan ke Pal_Dest dengan delay DeL.
Parameter : Pal_Dest = Alamat virtual palette tujuan.
Del = Delay loop.

Function : Metamorph_Palette (Pal_Buff, Pal_Dest : Palette_Addr; Rate:Byte) : Boolean;
Fungsi : Mengubah intensitas Pal_Buff ke Pal_Dest sebanyak Rate.
Parameter : Pal_Buff = Virtual palette yang akan diubah.
Pal_Dest = Virtual palette tujuan.
Rate = Jumlah perubahan yang dilakukan.
Return : TRUE jika seluruh intensitas Pal_Buff telah sama dengan Pal_Dest.

Procedure Scroll Layar

Procedure : Load_Vert_Scroll (Name:String);
Fungsi : Membaca background page yang akan dipergunakan untuk vertikal scrolling.
Parameter : Name = Nama file background page.

Procedure : Free_Vert_Scroll;
Fungsi : Melenyapkan background page.

Procedure : Put_Vert_Scroll (Var PicBuff; Y:Integer);
Fungsi : Meletakkan verticall scroll pada virtual screen.
Parameter : PicBuff = Alamat virtual screen.
Y = Koordinat Y sprite background page sebagai start awal dari penggambaran vertikal scroll.

Procedure : Load_Horiz_Scroll (Name:String);
Fungsi : Membaca background page yang akan dipergunakan untuk horizontal scrolling.
Parameter : Name = Nama file background page.

Procedure : Free_Horiz_Scroll;
Fungsi : Melenyapkan background page.

Procedure : Put_Horiz_Scroll (Var PicBuff;Y:Integer);
Fungsi : Meletakkan horizontal scroll pada virtual screen.

PicBuff = Alamat virtual screen.
Y = Koordinat X sprite background page sebagai start awal penggambaran horizontal scroll.

Procedure : Load_Parallax_Scroll (Name:String);
Fungsi : Membaca background page yang akan dipergunakan untuk parallax scrolling.
Parameter : Name = Nama file background page.

Procedure : Put_Parallax_Scroll (X,Y:Integer;Var PicBuff;Var Temp);
Fungsi : Meletakkan parallax scroll pada virtual screen.
Parameter : X,Y = Koordinat X dan Y sprite background page sebagai start awal penggambaran parallax scroll.
PicBuff = Alamat virtual screen.
Temp = Alamat memori tempat dilakukannya proses un-kompres oleh proces parallaxing.

Procedure Kompresi Sprite

Procedure : Color_Compress(Var Source;Var Destination;Var Size:Word);
Fungsi : Mengkompres data dengan metode Color Compress.
Parameter : Source = Alamat data yang akan dikompres.
Destination = Alamat tujuan.
Size = Output besar data setelah dikompres.

Procedure : Color_UnCompress (Var Source;Var Destination);
Fungsi : Meng-un-kompres data dengan metode Color Compress.
Parameter : Source = Alamat data yang akan di-un-kompres.
Destination = Alamat tujuan.

Procedure : Zero_Compress (Var Source;Var Destination;Var Size:Word);
Fungsi : Mengkompres data dengan metode Zero Compress.
Parameter : Source = Alamat data yang akan dikompres.
Destination = Alamat tujuan.
Size = Output besar data setelah dikompres.

Procedure : Zero_UnCompress (Var Source;Var Destination);
Fungsi : Meng-un-kompres data dengan metode Zero Compress.
Parameter : Source = Alamat data yang akan di-un-kompres.
Destination = Alamat tujuan.

Procedure : Rll_Compress (Var Source;Var Destination;Var Size:Word);
Fungsi : Mengkompres data dengan metode RLL Compress.
Parameter : Source = Alamat data yang akan dikompres.
Destination = Alamat tujuan.
Size = Output besar data setelah dikompres.

Procedure : RLL_UnCompress (Var Source,Destination);
Fungsi : Meng-un-kompres data dengan metode RLL Compress.
Parameter : Source = Alamat data yang akan di-un-kompres.
Destination = Alamat tujuan.

Procedure Pembantu

Procedure : GetAddr (Var PicBuff;Var Seg,Offs:Word);
Fungsi : Mengetahui alamat offset dan segment dari sebuah alamat pointer.
Parameter : PicBuff = Alamat pointer yang akan diperiksa.
Seg = Output Segment dari alamat pointer tersebut.
Offs = Output Offset dari alamat pointer tersebut.

BAB 7

PEMBUATAN GAME BUG HUNTER

Akhirnya sampailah kita pada bab yang ditunggu-tunggu. Pada bab ini kita akan membahas pembuatan game bertipe SUPER MARIO BROSS dan JACK JAZZ RABBIT dengan bermodalkan teknik-teknik yang telah kita bahas pada bab-bab sebelumnya.

Seperti pada pembuatan FIRE FIGHTER, proyek pembuatan game ini akan dibagi menjadi dua macam yaitu pembuatan game editor untuk merancang level dari game yang akan kita buat dan yang kedua ialah game engine yang berfungsi untuk menjalankan level yang dirancang oleh game editor.

Game ini akan kita beri nama BUG HUNTER. Pertama-tama kita akan merancang spesifikasi dari game ini.

PERENCANAAN

Game BUG HUNTER ini adalah game bertipe SUPER MARIO BROSS dengan memanfaatkan parallax scrolling. Player yang akan kita jalankan adalah seorang tokoh kartun. Adapun sudut pandang permainan ini ialah dari samping (side view). Selama permainan, layar akan digeser mengikuti arah player bergerak sehingga kita tidak akan kehilangan player dari layar.

Tokoh kartun yang akan kita mainkan akan kita beri nama BUGGY.

Adapun parallax scrolling yang akan kita gunakan ialah sebesar 3000 x 500 pixel. WOW besar sekali bukan? Inilah keajaiban yang diberikan oleh teknik RLL Compression. Tanpa fasilitas ini kita tidak akan dapat memperoleh background sebesar itu. Musuh-musuh akan diletakkan tersebar di seluruh layar. Juga balok-balok kecil dengan gambar yang bervariasi akan diletakkan tersebar di seluruh layar. Balok ini dapat dinaiki oleh setiap objek.

Karena jumlah musuh dan balok bisa sampai ratusan maka kita tidak akan memproses semuanya pada setiap loop, karena hal itu akan menimbulkan kecepatan permainan menjadi pelan sekali. Bayangkan kita harus mendeteksi tabrakan antara 200 objek terhadap 600 balok sehingga kita memerlukan total pendeteksian sekitar 200 x 600 = 120000 kali ditambah lagi dengan pendeteksiah tabrakan antara tiap objek, belum lagi ditambah dengan tabrakan peluru dengan balok & objek. Semuanya bisa-bisa memerlukan total pendeteksian sekitar 1 juta kali setiap loop. Perlu Anda ingat bahwa teknik pendeteksian kita menggunakan pembagian dan juga bilangan real yang memakan banyak cycle. Jika kita tetap bersikeras untuk memproses semua data yang ada bisa-bisa kecepatan game kita menjadi 1 frame per detik. Hal itu tentu saja sangat buruk.

Dengan sedikit akal licik, kita hanya akan memproses objek dan balok yang dekat dengan si Buggy. Definisi dekat ini kira-kira sekitar radius 200 pixel dari si Buggy. Dengan cara ini kita tidak perlu memproses semua objek.

BUGGY

- Dapat digerakkan secara horizontal dengan memanfaatkan teknik momentum.
- Dapat melompat dengan menekan tombol atas. Di sini akan berlaku teknik perhitungan gravitasi.
- Pada saat melompat, jika menabrak tembok maka kecepatan horizontal akan dihentikan.
- Pada saat melompat, jika menabrak atap maka BUGGY akan terpantul ke bawah.
- Pada saat melompat, jika kakinya menginjak balok maka BUGGY akan berdiri di atas balok tersebut.
- Buggy dapat membunuh musuh-musuh tertentu dengan cara menginjaknya dari atas.
- Buggy memiliki nyawa 6 buah dan setiap kali mati (tertabrak/tertembak musuh) maka nyawa akan berkurang 1. Tiga detik setelah kehilangan nyawa, si Buggy tidak akan dapat diserang oleh musuh. Untuk itu selama tiga detik ini si Buggy digambar dengan teknik transparant.
- Buggy akan berubah menjadi Super Buggy jika mengambil misile launcher. Dalam hal ini gambar dari si Buggy akan berubah menjadi Buggy yang berkaca mata hitam dan membawa misile launcher.
- Gambar dari Buggy ialah animasi berulang dari beberapa gambar dan untuk setiap gambar akan disediakan dua buah kopi yaitu gambar yang menghadap ke kiri dan gambar yang menghadap ke kanan.
- Buggy yang telah berubah menjadi Super Buggy dapat menembakkan misile ke arah musuh dan musuh yang kena langsung mati. Misile launcher ini akan hilang jika Buggy kehilangan nyawa.
- Misile launcher ini memiliki peluru yang akan diisi jika Buggy mengambil bonus rocket.
- Buggy dapat melompat lagi bila dia jatuh (selama tombol lompat belum ditekan).

- Buggy dapat memiliki kekebalan dan dapat membunuh musuh dengan cara menabraknya jika mendapat bonus kekebalan. Kekebalan ini hanya berlaku selama 10 detik.
- Buggy memiliki score dan score ini akan bertambah jika ia mengambil koin.

BINATANG MUSUH

Musuh ini terdiri atas 8 macam yang memiliki ciri-ciri khas tersendiri.

Karena background yang akan kita pergunakan sebesar 3000 x 500 pixel, maka tentu saja musuh:

- Shower Man
  - Bergerak secara horizontal, jika pada saat bergerak lantai tempat berpijak telah habis maka dia akan berganti arah.
  - Menembak setiap 4 detik ke arah si Buggy. Tembakannya ialah gelembung sabun.
  - Dapat dibunuh dengan cara diinjak kepalanya.
- Cloud
  - Bergerak secara horizontal di udara ke arah si Buggy.
  - Menembak setiap 5 detik secara vertikal yaitu dengan menjatuhkan air liur.
  - Dapat dibunuh dengan cara diinjak kepalanya.
- Teeth
  - Diam di tempat dan melompat ke arah si Buggy setiap 4 detik.
  - Dapat dibunuh dengan cara diinjak kepalanya.
- Peer Doll
  - Selalu melompat-lompat dan jika menabrak tembok ia akan berganti arah.
  - Dapat dibunuh dengan cara diinjak kepalanya.
- Red Wheel
  - Bergerak secara horizontal mengikuti permukaan tanah dan akan berganti arah jika menabrak tembok.
  - Tidak dapat dibunuh dengan cara diinjak.

- Piranha Plant
    - Tidak bergerak, tetapi masuk keluar dari dalam tanah.
    - Pada saat terlihat muncul dari dalam tanah, dia melakukan penyerangan tetapi jika masih belum kelihatan, si Buggy dapat melewatinya.
    - Tidak bisa dibunuh dengan cara diinjak.

PELURU

- Peluru si Buggy akan selalu meninggalkan animasi asap pada ekornya.
- Peluru musuh ada beberapa macam:
- Air liur yang ditembakkan oleh Cloud. Memiliki arah vertikal dan akan menampilkan animasi air yang menetes di tanah jika telah menabrak suatu permukaan tanah.
- Busa sabun yang ditembakkan oleh Shower Man mengarah ke si Buggy pada saat dia ditembakkan. Busa ini akan memanggil animasi busa pecah jika menabrak balok.

ANIMASI

Animasi ini adalah beberapa gambar yang ditampilkan secara berurutan setiap loop permainan pada koordinat tertentu.

Animasi di sini ialah objek yang tidak aktif, artinya tidak diikutsertakan dalam pendeteksian tabrakan. Tujuannya ialah cuma menampilkan animasi pada layar. Animasi ini ada beberapa jenis:

- Percikan api tembakan si Buggy yang mengenai dinding.
- Ledakan merah jika Buggy membunuh musuh.
- Asap yang dihasilkan dari ekor roket si Buggy. Asap ini akan ditampilkan dengan teknik transparant.
- Tetesan air pada permukaan tanah (air liur dari si Cloud)
- Busa sabun yang meledak (tembakan si Shower Man)
- Bola kekebalan yang akan muncul di tengah-tengah si Buggy jika Buggy mendapat bonus kekebalan. Bola ini akan ditampilkan dengan teknik transparant.

TEXT

Text di sini hampir sama seperti animasi tetapi tujuannya ialah menampilkan tulisan yang bergerak ke atas. Dengan cara ini akan dapat menampilkan pesan-pesan. Misalnya setelah si buggy mengambil koin kita dapat menampilkan tulisan "100 PTS" yang berarti si buggy mendapat nilai 100 point.

BALOK

Balok di sini ialah gambar-gambar sprite yang berukuran 16 x 11 yang akan disusun di layar. Gunanya ialah sebagai perintang dan juga sebagai tempat berpijak bagi objek.

Nah dengan spesifikasi di atas kita akan merancang game BUG HUNTER ini.

PEMBUATAN MISSION EDITOR

Seperti yang telah dijelaskan pada halaman yang terdahulu, kita akan merancang EDITOR dari game ini. Kali ini Editor yang akan kita buat tidaklah serumit yang kita buat pada waktu merancang FIRE FIGHTER karena gerakan dari setiap musuh akan dikendalikan lewat engine dari BUG HUNTER ini. Jadi tugas dari Mission Editor hanyalah meletakkan balok-balok dan musuh-musuh pada layar sebesar 3000 x 500 pixel. Jangan lupa untuk meletakkan si Buggy dan cukup satu saja. Jangan meletakkan Buggy lebih dari satu pada layar permainan karena Buggy yang ditemukan pertama kali oleh engine akan dijadikan player Anda, sedangkan engine akan memaksa Buggy yang lain untuk dijadikan musuh.

Mission Editor kita kali ini akan memanfaatkan MOUSE sehingga proses editting akan jauh lebih mudah dibandingkan pada saat kita menggunakan FIRE FIGHTER MISSION EDITOR.

OK siap?? Inilah listing program dari BUG HUNTER MISSION EDITOR.

```
{-------------------------------------------}
{ Program: B_EDITOR.PAS                     }
{ Coder  : Agustinus Nalwan                 }
{ Fungsi : Editor dari Bug Hunter game.     }
{-------------------------------------------}

Uses Crt,VgaCad,Dos,VgaFont;

Const Spot     =  1;
      Shower   = 22;
      Piranha  = 20;
      Ghost    = 28;
      Cloud    = 33;
      Peer     = 36;
      Wheel    = 40;
      Snake    = 42;
      Teeth    = 44;

      P_Brown  = 47;
      P_Red    = 55;
      P_Green  = 57;
      P_Gray   = 58;
      M_Red    = 48;
      M_Green  = 49;
      M_Blue   = 50;
      H_Red    = 51;
      H_Brown  = 52;
      H_Gray   = 53;
      E_Box    = 56;
      Mold1    = 54;
      Mold2    = 59;

      Bomb     = 88;
      Misile   = 91;
      Coin     = 92;
      Invul    = 95;

Type Enemy_Rec = Record
       X,Y    : Integer;
       Tipe   : Byte;
       Energy : Integer;
     End;

     Block_Rec = Record
       X,Y    : Integer;
       Pic    : Byte;
     End;

     Char_Rec = Record
       Length,Width: Word;
       Picture     : Integer;
     End;
```

```
Var Fil          : File;
    Atr          : Text;
    A            : Integer;
    Picture      : Array[1..200] of Pointer;
    Finish       : Boolean;
    Ch           : Char;
    Szs          : Word;
    Objek_Char   : Array[1..20] of Char_Rec;
    Enemy_Spot   : Array[1..200] of  Enemy_Rec;
    Block        : Array[1..1000] of Block_Rec;
    Mission_File: File;
    Process      : Pointer;
    UnComp       : Pointer;
    Xp,Yp        : Integer;
    ToolBar      : Pointer;
    LastBt       : Byte;
    Xm,Ym        : Integer;
    Bt           : Byte;
    Same         : Boolean;
    Ctr          : Byte;
    ScSpeed      : Byte;
    IconTagged   : Integer;

Procedure Small;External;
{$l Small.Obj}

Procedure InstallMouse;
  Begin
    Reg.AX:=$1;
    Intr($33,Reg);
  End;

Procedure UnInstallMouse;
  Begin
    Reg.AX:=$2;
    Intr($33,Reg);
  End;

Procedure SetMouse(Xm,Ym:Integer);
  Begin
    Reg.Ax:=$4;
    Reg.Cx:=Xm*2;
    Reg.Dx:=Ym;
    Intr($33,Reg);
  End;

Procedure ReadMouse;
  Begin
    Reg.AX:=$3;
    Intr($33,Reg);
    Xm:=Reg.Cx div 2;
    Ym:=Reg.Dx;
    LastBt:=Bt;
    Bt:=Reg.Bx;
```

```
    If LastBt<>Bt then Same:=False else Same:=True;
    If Bt>1 then
      Begin
      End;
    Inc(Ctr);
  End;

Function Get_Picture(Icon:Integer):Integer;
  Begin
    Case Icon of
      1:Get_Picture:=Spot;
      2:Get_Picture:=Ghost;
      3:Get_Picture:=Shower;
      4:Get_Picture:=Piranha;
      5:Get_Picture:=Peer;
      6:Get_Picture:=Wheel;
      7:Get_Picture:=Cloud;
      8:Get_Picture:=Teeth;
      9:Get_Picture:=Snake;
      10:Get_Picture:=Bomb;
      11:Get_Picture:=Misile;
      12:Get_Picture:=Invul;
      13:Get_Picture:=Coin;
      21:Get_Picture:=P_Brown;
      22:Get_Picture:=M_Red;
      23:Get_Picture:=M_Green;
      24:Get_Picture:=M_Blue;
      25:Get_Picture:=H_Red;
      26:Get_Picture:=H_Brown;
      27:Get_Picture:=H_Gray;
      28:Get_Picture:=Mold1;
      29:Get_Picture:=P_Red;
      30:Get_Picture:=E_Box;
      31:Get_Picture:=P_Green;
      32:Get_Picture:=P_Gray;
      33:Get_Picture:=Mold2;
    End;
  End;
Procedure PutMouse(Var Process;Mode:Byte);
  Begin
    Case Mode of
      1:Begin
          Rectangle(Process,Xm-2,Ym-2,Xm+2,Ym+2,Ctr);
        End;
      2:Begin
          Line(Process,Xm,Ym-2,Xm,Ym+2,Ctr);
          Line(Process,Xm-2,Ym,Xm+2,Ym,Ctr);
        End;
      3:Begin
          PutPixel(Process,Xm,Ym,Ctr);
        End;
      4:Begin
          If IconTagged<41 then
            Begin
```

```
Rll_UnCompress(Picture[Get_Picture(IconTagged)]^,UnComp^);
                PutImage(Process,Xm,Ym,UnComp^);
              End;
            Line(Process,Xm,Ym-2,Xm,Ym+2,Ctr);
            Line(Process,Xm-2,Ym,Xm+2,Ym,Ctr);
          End;
      End;
      Inc(Ctr);
    End;

Procedure Get_Dimension(Var Pic;Var X,Y:Word);
Var Length,Width:Word;
  Begin
    Asm
      Push ds
      Les di, Pic
      Mov Ax,[ES:DI]
      Mov Length,Ax
      Add di,2
      Mov Ax,[ES:DI]
      Mov Width,Ax
      Pop ds
    End;
    X:=Length;
    Y:=Width;
  End;

Procedure Put(Var Proc;X,Y:Integer;Pic:Integer);
Var Xl,Yl:Word;
  Begin
    If (X<319) and (Y<199) and (X>-50) and (Y>-50) then
      Begin
        Rll_UnCompress(Picture[Pic]^,UnComp^);
        Get_Dimension(UnComp^,Xl,Yl);
        If (X+Xl>0) and (Y+Yl>0) then
          Begin
            PutImage(Proc,X,Y,UnComp^);
          End;
      End;
  End;

Procedure Preparation;
  Begin
    OpenVga256;
    LoadFont(@Small);
    Install_Palette('Palette.Pal');
    Load_Parallax_Scroll('Back1');
    GetMem(Process,64000);
    GetMem(UnComp,10000);
    GetMem(ToolBar,10000);
    FillChar(Block,5000,0);
    FillChar(Enemy_Spot,1400,0);
    IconTagged:=0;
    Object_Char[1].Picture:=Spot;
```

```
    Object_Char[2].Picture:=Ghost;
    Object_Char[3].Picture:=Shower;
    Object_Char[4].Picture:=Piranha;
    Object_Char[5].Picture:=Peer;
    Object_Char[6].Picture:=Wheel;
    Object_Char[7].Picture:=Cloud;
    Object_Char[8].Picture:=Teeth;
    Object_Char[9].Picture:=Snake;
    Object_Char[10].Picture:=Bomb;
    Object_Char[11].Picture:=Misile;
    Object_Char[12].Picture:=Invul;
    Object_Char[13].Picture:=Coin;
  End;

Procedure Init_Picture;
  Begin
    Assign(Fil,'PICS\TOOLBAR.PIC');
    Reset(Fil,1);
    Assign(Atr,'PICS\TOOLBAR.SZS');
    Reset(Atr);
    Readln(Atr,Szs);
    BlockRead(Fil,UnComp^,Szs);
    Close(Fil);
    Close(Atr);
    Rll_UnCompress(UnComp^,ToolBar^);
    Assign(Fil,'PICS\SPRITE.PIC');
    Reset(Fil,1);
    Assign(Atr,'PICS\SPRITE.SZS');
    Reset(Atr);
    Finish:=False;
    A:=1;
    Repeat
      Readln(Atr,Szs);
      GetMem(Picture[A],Szs);
      BlockRead(Fil,Picture[A]^,Szs);
      Inc(A);
    Until EOF(Atr);
    Close(Fil);
    Close(Atr);

    Assign(Fil,'PICS\SPRITE2.PIC');
    Reset(Fil,1);
    Assign(Atr,'PICS\SPRITE2.SZS');
    Reset(Atr);
    Repeat
      Readln(Atr,Szs);
      GetMem(Picture[A],Szs);
      BlockRead(Fil,Picture[A]^,Szs);
      Inc(A);
    Until EOF(Atr);
    Close(Fil);
    Close(Atr);
    For A:=1 to 13 do
      Begin
```

```
        Rll_UnCompress(Picture[Object_Char[A].Picture]^,UnComp^);

Get_Dimension(UnComp^,Object_Char[A].Length,Object_Char[A].Width);
      End;
  End;

Procedure Put_Display;
Var A:Integer;
  Begin
    PutImage(Process^,5,172,ToolBar^);
    If IconTagged<>0 then
      Begin
        If IconTagged<20 then
          Rectangle(Screen^,5+(IconTagged-
1)*21,172,3+IconTagged*21,187,15) else
        If IconTagged<40 then
          Rectangle(Screen^,5+(IconTagged-21)*16,188,;
                    3+(IconTagged-20)*16,199,15) else
        If IconTagged<60 then
          Rectangle(Screen^,213+(IconTagged-41)*25,188,;
                    211+(IconTagged-40)*25,199,15);
      End;
    For A:=1 to 200 do
      With Enemy_Spot[A] do
        If Tipe<>0 then
          Begin
            Put(Process^,X-Xp,Y-Yp,Object_Char[Tipe].Picture);
          End;
    For A:=1 to 1000 do
      With Block[A] do
        If Pic<>0 then
          Begin
            Put(Process^,X-Xp,Y-Yp,Pic);
          End;
  End;

Function Search_Object(Var Tip:Integer;Xm,Ym:Integer):Integer;
Var A:Integer;
  Begin
    Search_Object:=0;
    For A:=1 to 200 do
      With Enemy_Spot[A] do
        If Tipe<>0 then
          Begin
            If (Xm+Xp>X) and (Ym+Yp>Y) and;
                (Xm+Xp<X+Object_Char[Tipe].Length) and
            (Ym+Yp<Y+Object_Char[Tipe].Width) then
              Begin
                Tip:=Tipe;
                Search_Object:=A;
                A:=200;
              End;
          End;
  End;
```

```
Function Search_Block(Var Tip:Integer;Xm,Ym:Integer):Integer;
Var A:Integer;
  Begin
    Search_Block:=0;
    For A:=1 to 1000 do
      With Block[A] do
        If Pic<>0 then
          Begin
            If (Xm+Xp>X) and (Ym+Yp>Y) and (Xm+Xp<X+16) and
            (Ym+Yp<Y+11) then
              Begin
                Case Pic of
                  P_Brown:Tip:=21;
                  M_Red :Tip:=22;
                  M_Green:Tip:=23;
                  M_Blue:Tip:=24;
                  H_Red :Tip:=25;
                  H_Brown:Tip:=26;
                  H_Gray:Tip:=27;
                  Mold1 :Tip:=28;
                  P_Red :Tip:=29;
                  E_Box :Tip:=30;
                  P_Green:Tip:=31;
                  P_Gray:Tip:=32;
                  Mold2 :Tip:=33;
                End;
                Search_Block:=A;
                A:=1000;
              End;
          End;
  End;

Procedure Move_Object;
Var Tipe,Obj:Integer;
  Begin
    Obj:=Search_Object(Tipe,Xm,Ym);
    If Obj=0 then
      Begin
        Obj:=Search_Block(Tipe,Xm,Ym);
        If Obj<>0 then
          Begin
            Block[Obj].Pic:=0;
            IconTagged:=Tipe;
          End;
      End Else
      Begin
        IconTagged:=Tipe;
        Enemy_Spot[Obj].Tipe:=0;
      End;
  End;

Procedure Delete_Object;
Var Tipe,Obj:Integer;
  Begin
```

```
    Obj:=Search_Object(Tipe,Xm,Ym);
    if Obj=0 then
      Begin
        Obj:=Search_Block(Tipe,Xm,Ym);
        If Obj<>0 then
          Block[Obj].Pic:=0;
      End Else
      Begin
        Enemy_Spot[Obj].Tipe:=0;
      End;
  End;

Function Get_Object_Tipe(No:Integer):Integer;
  Begin
    Case No of
      1..13:Get_Object_Tipe:=No;
    End;
  End;

Procedure Save_Mission;
  Begin
    Assign(Mission_File,'Pics\Mission.001');
    Rewrite(Mission_File,1);
    BlockWrite(Mission_File,Object_Char,20*SizeOf(Char_Rec));
    BlockWrite(Mission_File,Enemy_Spot,200*SizeOf(Enemy_Rec));
    BlockWrite(Mission_File,Block,1000*SizeOf(Block_Rec));
    Close(Mission_File);
  End;

Procedure Put_Center(Y:Integer;Col1,Col2:Byte;St:String);
  Begin
    PutString(Screen^,150-(Length(St) div 2)*6,Y,Col1,Col2,St);
  End;

Procedure Show_Credit;
  Begin
    VgaCad.Block(Screen^,40,40,270,130,8);
    Rectangle(Screen^,40,40,270,130,7);
    Put_Center(50,0,15,'BUG HUNTER MISSION EDITOR');
    Put_Center(70,0,14,'CREATED BY AGUSTINUS NALWAN');
    Put_Center(90,0,11,'APROMPT ENTERTAINMENT DIVISION');
    Put_Center(110,0,12,'COPYRIGHT SEPTEMBER 28 1995');
    Ch:=Readkey;
  End;

Procedure Detect_Mouse_Command;
Var Opt:Integer;
  Begin
    If (Bt=1) and Not Same then
      Begin
        If Ym>=172 then
          Begin
            If Ym<188 then
              Begin
```

```
        If Xm<278 then
          Begin
            Opt:=((Xm-5) div 21) + 1;
            IconTagged:=Opt;
          End;
      End Else
      Begin
        If Xm<213 then
          Begin
            Opt:=((Xm-5) div 16) + 21;
            IconTagged:=Opt;
          End Else
          Begin
            Opt:=((Xm-209) div 25) + 41;
            IconTagged:=Opt;
          End;
      End;
    If (Xm>278) and (Xm<313) and (Ym<189) then
      Begin
        Show_Credit;
        IconTagged:=0;
      End;
  End Else
  Begin
      If (IconTagged>20) and (IconTagged<41) then
        Begin
          For A:=1 to 1000 do
            With Block[A] do
              If Pic=0 then
                Begin
                  Pic:=Get_Picture(IconTagged);
                  X:=Xm+Xp;Y:=Ym+Yp;
                  A:=1000;
                End;
        End Else
      If (IconTagged<>0) and (IconTagged<41) then
        Begin
          For A:=1 to 200 do
            With Enemy_Spot[A] do
              If Tipe = 0 then
                Begin
                  X:=Xm+Xp;Y:=Ym+Yp;
                  Tipe:=Get_Object_Tipe(IconTagged);
                  A:=200;
                End;
        End Else
        Case IconTagged of
          42:Move_Object;
          43:Delete_Object;
        End;
   End;
Case IconTagged of
  41:Save_Mission;
  44:Begin
```

```
                Sound(100);Delay(30);NoSound;
                VgaCad.Block(Screen^,80,70,240,90,8);
                VgaCad.Rectangle(Screen^,80,70,240,90,7);
                Put_Center(80,0,15,'EXIT EDITTING (Y/N)?');
                Ch:=Readkey;
                If (Ch='y') or (Ch='Y') then Finish:=True;
              End;
          End
        End;
  End;

Procedure Main;
  Begin
    Finish:=False;
    Xp:=0;Yp:=0;
    ScSpeed:=0;
    Repeat
      Repeat
        Put_Parallax_Scroll(Xp,Yp,Process^,UnComp^);
        Put_Display;
        ReadMouse;
        Detect_Mouse_Command;
        If IconTagged=0 then
          PutMouse(Process^,2) else
          PutMouse(Process^,4);
        MoveTo(Process^,Screen^);
        If (Xm=319) or (Xm=0) or (Ym=199) or (Ym=0) then
          Inc(ScSpeed)
        Else
          ScSpeed:=0;
        If ScSpeed>50 then ScSpeed:=50;
        If Xm=319 then Inc(Xp,ScSpeed);
        If Xm=0 then Dec(Xp,ScSpeed);
        If Ym=199 then Inc(Yp,ScSpeed);
        If Ym=0 then Dec(Yp,ScSpeed);
      Case Ch of
        #72:Dec(Yp,1);
        #80:Inc(Yp,1);
        #75:Dec(Xp,1);
        #77:Inc(Xp,1);
      End;
      If Xp<0 then Xp:=0;
      If Yp<0 then Yp:=0;
      If Xp>2680 then Xp:=2680;
      If Yp>300 then Yp:=300;
      Until Keypressed or Finish;
      If Not Finish then Ch:=Readkey;
    Until Finish;
  End;

Procedure Init_Mission;
  Begin
    Assign(Mission_File,'Pics\Mission.001');
    Reset(Mission_File,1);
```

```
    BlockRead(Mission_File,Object_Char,20*SizeOf(Char_Rec));
    BlockRead(Mission_File,Enemy_Spot,200*SizeOf(Enemy_Rec));
    BlockRead(Mission_File,Block,1000*SizeOf(Block_Rec));
    Close(Mission_File);
  End;

Procedure Removal;
  Begin
    CloseVga256;
  End;

Begin
  Preparation;
  Init_Picture;
  Init_Mission;
  Main;
  Removal;
End.
```

CARA PEMAKAIAN BUG HUNTER MISSION EDITOR

Ingat jalankan MOUSE driver terlebih dahulu sebelum menjalankan EDITOR ini.

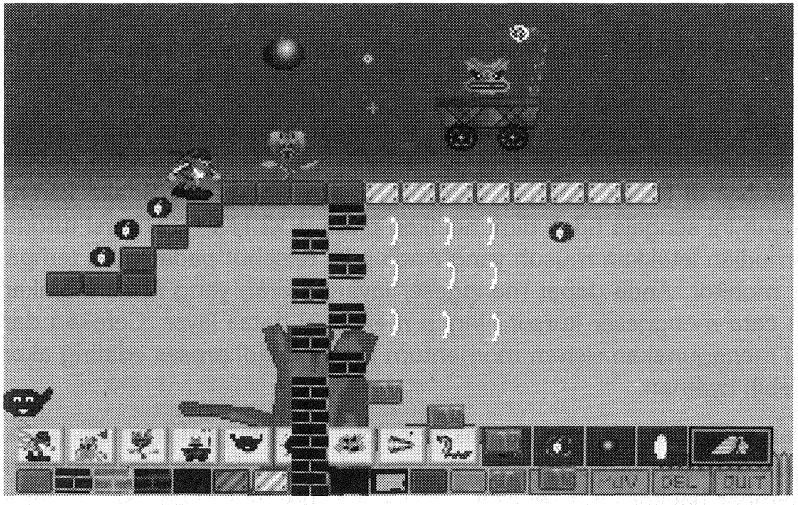
Sebenarnya cara pemakaian EDITOR ini sangatlah sederhana. Untuk meletakkan sebuah musuh, Anda tinggal memilih salah satu dari icon musuh pada gambar di bawah (jangan memilih musuh Ghost dan Snake karena dua musuh ini sengaja tidak dimunculkan pada BUG HUNTER versi tutorial ini). Kemudian letakkan musuh yang Anda pilih tersebut pada suatu tempat di layar monitor. Anda dapat menggeser layar ke arah yang Anda inginkan dengan cara menggeser kursor hingga mencapai batas layar. Gerakan dari layar ini menggunakan sistem momentum juga lho jadi proses editting akan terasa lebih nyaman.

Demikian pula dengan cara peletakan balok. Anda tinggal memilih balok, kemudian letakkan di salah satu tempat pada layar. Bila Anda perhatikan pada gambar background sebesar 3000 x 500 pixel ini Anda akan mendapati permukaan tanah di bagian bawah. Permukaan tanah ini nantinya secara otomatis akan dianggap sebagai balok oleh engine BUG HUNTER sehingga kita tidak perlu meletakkan balok pada permukaan tanah tersebut sebagai tempat berpijak. Yang perlu Anda ingat ialah daerah lain yang tidak ada baloknya dan bukan merupakan permukaan tanah dianggap sebagai langit kosong, sehingga jika ada objek berada pada bagian tersebut, objek tersebut akan jatuh.

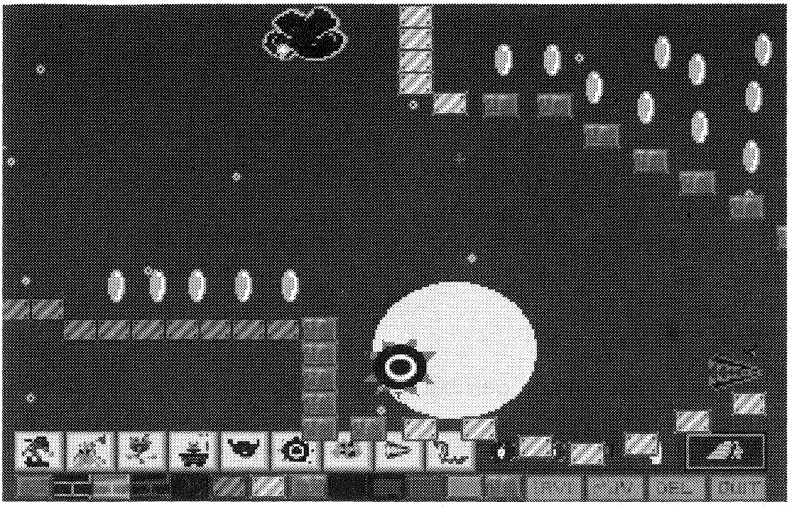
Satu lagi yang perlu diperhatikan ialah jangan meletakkan objek bertabrakan dengan permukaan tanah atau bertabrakan dengan balok, karena akibatnya objek tersebut tidak akan dapat bergerak.

Editor ini dilengkapi pula dengan fasilitas DEL yang berfungsi untuk menghapus sebuah balok atau objek pada layar monitor. Caranya ialah dengan memilih icon DEL ini kemudian pilih objek atau balok yang akan Anda hapus. Di samping itu disediakan pula fasilitas MOV yang bertujuan untuk memindahkan objek/balok pada layar monitor. Caranya ialah dengan memilih icon MOV kemudian pilih objek/balok pada layar yang akan Anda pindahkan. Kemudian setelah objek/balok tersebut terpilih, pindahkan ke posisi yang lain.

Setelah proses editting level selesai, pilihlah icon SAVE yang bertujuan untuk menyimpan level yang telah Anda rancang. Ingat bahwa semua level yang Anda rancang akan disimpan ke dalam file yang sama sehingga hati-hati untuk menumpuk level yang telah Anda rancang sebelumnya.



Gambar 7.1 Screen Shoot dari BUG HUNTER Mission Editor



Gambar 7.2 Screen Shoot dari BUG HUNTER Mission Editor

Demikianlah cara pemakaian BUG HUNTER MISSION EDITOR. Sederhana sekali bukan?

PENJELASAN PROGRAM

```
Uses Crt,VgaCad,Dos,VgaFont;

Const Spot    =  1;
      Shower  = 22;
      Piranha = 20;
      Ghost   = 28;
      Cloud   = 33;
      Peer    = 36;
      Wheel   = 40;
      Snake   = 42;
      Teeth   = 44;
```

Konstanta di atas ialah nomor gambar yang digunakan pada saat musuh yang bersangkutan Anda pilih.

Dengan cara ini kita tidak perlu mengingat nomor gambar dari tiap musuh.

```
      P_Brown = 47;
      P_Red   = 55;
      P_Green = 57;
      P_Gray  = 58;
      M_Red   = 48;
      M_Green = 49;
      M_Blue  = 50;
      H_Red   = 51;
```

```
H_Brown = 52;
H_Gray  = 53;
E_Box   = 56;
Mold1   = 54;
Mold2   = 59;
```

Konstanta di atas ialah nomor gambar yang digunakan, pada saat balok yang bersangkutan Anda pilih.

```
Bomb    = 88;
Misile  = 91;
Coin    = 92;
Invul   = 95;
```

Konstanta di atas ialah nomor gambar yang digunakan pada saat bonus yang bersangkutan Anda pilih.

```
Type Enemy_Rec = Record
       X,Y    : Integer;
       Tipe   : Byte;
       Energy : Integer;
     End;
```

Record di atas ialah struktur data dari objek yang akan kita gunakan.

- X ialah koordinat X dari obejct.
- Y ialah koordinat Y dari objek.
- Tipe ialah jenis objek (membedakan musuh yang satu dan yang lain, juga untuk membedakan bonus).
- Energy tidak terpakai.

```
Block_Rec = Record
  X,Y    : Integer;
  Pic    : Byte;
End;
```

Record di atas ialah struktur data dari balok yang kita gunakan.

- X ialah koordinat X dari balok.
- Y ialah koordinat Y dari balok.
- Pic ialah nomor gambar yang dipergunakan oleh balok.

```
Char_Rec = Record
  Length,Width: Word;
  Picture     : Integer;
End;
```

Record di atas digunakan untuk menyimpan data gambar objek yaitu lebar, panjang dan nomor gambar.

```
Var Fil         : File;
    Atr         : Text;
    A           : Integer;
    Picture     : Array[1..200] of Pointer;
```

Array of Pointer untuk menampung data gambar permainan.

```
    Finish      : Boolean;
```

Untuk mengakhiri game.

```
    Ch          : Char;
    Szs         : Word;
    Object_Char : Array[1..20] of Char_Rec;
```

Objek_Char adalah variable array of yang berisi data gambar dari setiap objek.

```
    Enemy_Spot : Array[1..200] of  Enemy_Rec;
```

Array yang digunakan untuk menampung data objek.

```
    Block       : Array[1..1000] of Block_Rec;
```

Array untuk menampung data tiap balok.

```
    Mission_File: File;
```

Variabel file untuk menyimpan level yang kita rancang.

```
    Process     : Pointer;
    UnComp      : Pointer;
```

Variabel pointer untuk menampung virtual screen dan untuk membuka data yang dikompres.

```
    Xp,Yp       : Integer;
```

Koordinat parallax scrolling.

```
    ToolBar     : Pointer
```

Variabel pointer untuk menyimpan gambar icon EDITOR.

```
    LastBt        : Byte;
    Xm,Ym       : Integer;
```

Xm dan Ym digunakan untuk menyimpan koordinat MOUSE.

LastBt digunakan untuk menyimpan tombol yang lalu yang telah ditekan.

```
    Bt              : Byte;
    Same         : Boolean;
```

Variabel Bt digunakan untuk menyimpan kondisi tombol mouse yang disimpan. Sedangkan Same adalah variabel yang menunjukkan kalau tombol mouse yang sebelumnya ditekan sama dengan tombol mouse yang sekarang.

```
    Ctr                : Byte;
    ScSpeed         : Byte;
    IconTagged : Integer;
```

ScSpeed adalah kecepatan parallax scroll. Variabel ini kita gunakan untuk menghasilkan efek momentum pada proses scrolling.

IconTagged adalah variabel yang menunjukkan nomor icon yang sedang dipilih.

```
Procedure Small;External;
{$l Small.Obj}
```

Compile procedure luar untuk mengakses font.

```
Procedure InstallMouse;
  Begin
    Reg.AX:=$1;
    Intr($33,Reg);
  End;
```

Lakukan instalasi mouse yaitu dengan memanggil interrupt $33 servis 1.

```
Procedure UnInstallMouse;
  Begin
    Reg.AX:=$2;
    Intr($33,Reg);
  End;
```

Lakukan uninstalasi mouse yaitu dengan memanggil interrupt $33 servis 2.

```
Procedure SetMouse(Xm,Ym:Integer);
  Begin
    Reg.Ax:=$4;
    Reg.Cx:=Xm*2;
    Reg.Dx:=Ym;
    Intr($33,Reg);
  End;
```

Paksa kursor mouse ke koordinat (Xm,Ym). Di sini Xm dikalikan 2 karena koordinat mouse lain dengan koordinat layar. Koordinat mouse ialah 640 x 200.

```
Procedure ReadMouse;
  Begin
    Reg.AX:=$3;
    Intr($33,Reg);
    Xm:=Reg.Cx div 2;
    Ym:=Reg.Dx;
    LastBt:=Bt;
    Bt:=Reg.Bx;
    If LastBt<>Bt then Same:=False else Same:=True;
    If Bt>1 then
      Begin
      End;
    Inc(Ctr);
  End;
```

Baca koordinat kursor dari mouse. Koordinat ini akan dimasukkan ke dalam variabel Xm dan Ym. Kemudian simpan tombol terakhir yang ditekan ke dalam LastBt dan ambil tombol baru yang ditekan ke dalam variabel Bt.

Jika LastBt = Bt maka tombol yang lama dan tombol sekarang adalah sama. Ini menunjukkan bahwa tombol mouse sedang ditekan dan belum dilepas.

```
Function Get_Picture(Icon:Integer):Integer;
  Begin
    Case Icon of
      1:Get_Picture:=Spot;
      2:Get_Picture:=Ghost;
      3:Get_Picture:=Shower;
      4:Get_Picture:=Piranha;
      5:Get_Picture:=Peer;
      6:Get_Picture:=Wheel;
      7:Get_Picture:=Cloud;
      8:Get_Picture:=Teeth;
      9:Get_Picture:=Snake;
      10:Get_Picture:=Bomb;
      11:Get_Picture:=Misile;
      12:Get_Picture:=Invul;
      13:Get_Picture:=Coin;
      21:Get_Picture:=P_Brown;
      22:Get_Picture:=M_Red;
      23:Get_Picture:=M_Green;
      24:Get_Picture:=M_Blue;
      25:Get_Picture:=H_Red;
      26:Get_Picture:=H_Brown;
      27:Get_Picture:=H_Gray;
      28:Get_Picture:=Mold1;
      29:Get_Picture:=P_Red;
```

```
      30:Get_Picture:=E_Box;
      31:Get_Picture:=P_Green;
      32:Get_Picture:=P_Gray;
      33:Get_Picture:=Mold2;
    End;
  End;
```

Fungsi ini digunakan untuk menunjukkan nomor gambar dari icon yang dikirimkan lewat parameter.

Caranya ialah dengan menggunakan konstanta di atas.

```
Procedure PutMouse(Var Process;Mode:Byte);
  Begin
    Case Mode of
      1:Begin
          Rectangle(Process,Xm-2,Ym-2,Xm+2,Ym+2,Ctr);
        End;
      2:Begin
          Line(Process,Xm,Ym-2,Xm,Ym+2,Ctr);
          Line(Process,Xm-2,Ym,Xm+2,Ym,Ctr);
        End;
      3:Begin
          PutPixel(Process,Xm,Ym,Ctr);
        End;
      4:Begin
          If IconTagged<41 then
            Begin

Rll_UnCompress(Picture[Get_Picture(IconTagged)]^,UnComp^);
              PutImage(Process,Xm,Ym,UnComp^);
            End;
          Line(Process,Xm,Ym-2,Xm,Ym+2,Ctr);
          Line(Process,Xm-2,Ym,Xm+2,Ym,Ctr);
        End;
    End;
    Inc(Ctr);
  End;
```

Procedure di atas berguna untuk menampilkan kursor mouse. Jika Mode yang dikirimkan berharga 3 maka bentuk kursor ialah titik, jika mode berharga 4 maka kursor yang ditampilkan ialah gambar objek yang sedang dipilih.

```
Procedure Get_Dimension(Var Pic;Var X,Y:Word);
Var Length,Width:Word;
  Begin
    Asm
```

Procedure ini bertujuan untuk mengambil panjang dan lebar sprite yang disimpan pada empat byte pertama dari sprite.

```
    Push ds
      Les di, Pic
      Mov Ax,[ES:DI]
      Mov Length,Ax
```

Baca panjang sprite.

```
      Add di,2
      Mov Ax,[ES:DI]
      Mov Width,Ax
```

Baca lebar sprite.

```
      Pop ds
    End;
    X:=Length;
    Y:=Width;
```

Keluarkan panjang dan lebar ke parameter.

```
  End;

Procedure Put(Var Proc;X,Y:Integer;Pic:Integer);
Var Xl,Yl:Word;
  Begin
```

Procedure ini berguna untuk meletakkan sebuah sprite pada layar monitor. Dengan cara ini kita tidak perlu melakukan proses un-kompres kemudian melakukan proses peletakan gambar ke layar setiap kali kita hendak meletakkan sebuah sprite pada layar.

```
    If (X<319) and (Y<199) and (X>-50) and (Y>-50) then
      Begin
```

Deteksi apakah gambar tersebut masih dalam batas pandangan layar.

```
        Rll_UnCompress(Picture[Pic]^,UnComp^);
```

Lakukan proses un-kompres.

```
        Get_Dimension(UnComp^,Xl,Yl);
```

Ambil panjang dan lebar gambar.

```
        If (X+Xl>0) and (Y+Yl>0) then
```

Lakukan pendeteksian kedua yaitu dengan mendeteksi batas kanan gambar. Tujuan dilakukannya sistem pendeteksian bertahap ini ialah untuk optimasi kecepatan karena dengan memanggil procedure Get_Dimension maka kecepatan animasi akan berkurang sedikit. Sedangkan untuk melakukan pendeteksian batas kanan gambar diperlukan panjang dan lebar gambar. Untuk

itu pada testing pertama di atas kita gunakan harga -50 sebagai harga perkiraan karena tidak ada sprite yang lebih besar dari 50 pixel.

```
          Begin
            PutImage(Proc,X,Y,UnComp^);
```

Letakkan sprite pada virtual screen.

```
          End;
      End;
  End;
Procedure Preparation;
  Begin
    OpenVga256;
    LoadFont(@Small);
    Install_Palette('Palette.Pal');
    Load_Parallax_Scroll('Back1');
```

Load Font yang akan kita pakai. Instal Palette dari file 'PALETTE.PAL' kemudian load background page untuk parallax scrolling dari file Back1.SCR.

```
    GetMem(Process,64000);
    GetMem(UnComp,10000);
    GetMem(ToolBar,10000);
```

Sediakan memori untuk virtual page, proses un-kompres gambar dan untuk toolbar.

```
    FillChar(Block,5000,0);
    FillChar(Enemy_Spot,1400,0);
```

Isikan variabel array yang digunakan untuk menampung data objek dan balok dengan 0. Dengan ini diasumsikan bahwa masih belum ada balok dan objek di layar.

```
    IconTagged:=0;
```

Asumsikan belum ada icon yang dipilih.

```
    Object_Char[1].Picture:=Spot;
    Object_Char[2].Picture:=Ghost;
    Object_Char[3].Picture:=Shower;
    Object_Char[4].Picture:=Piranha;
    Object_Char[5].Picture:=Peer;
    Object_Char[6].Picture:=Wheel;
    Object_Char[7].Picture:=Cloud;
    Object_Char[8].Picture:=Teeth;
    Object_Char[9].Picture:=Snake;
    Object_Char[10].Picture:=Bomb;
    Object_Char[11].Picture:=Misile;
    Object_Char[12].Picture:=Invul;
    Object_Char[13].Picture:=Coin;
```

Isikan nomor gambar dari tiap objek ke dalam array Objek_Char. Sehingga kita dapat menentukan gambar dari sebuah objek dengan menggunakan variabel Objek_Char[Tipe Objek].Picture. Setiap jenis objek memiliki tipe objek yang berbeda seperti Buggy memiliki nomor kode 1, Cloud memiliki nomor kode 7. Buggy pada program ini digantikan dengan nama SPOT.

```
  End;

Procedure Init_Picture;
  Begin
    Assign(Fil,'PICS\TOOLBAR.PIC');
    Reset(Fil,1);
    Assign(Atr,'PICS\TOOLBAR.SZS');
    Reset(Atr);
    Readln(Atr,Szs);
    BlockRead(Fil,UnComp^,Szs);
    Close(Fil);
    Close(Atr);
```

Baca gambar toolbar file.

```
    Rll_UnCompress(UnComp^,ToolBar^);
```

Lakukan proses un-kompress.

```
    Assign(Fil,'PICS\SPRITE.PIC');
    Reset(Fil,1);
    Assign(Atr,'PICS\SPRITE.SZS');
    Reset(Atr);
    Finish:=False;
    A:=1;
    Repeat
      Readln(Atr,Szs);
      GetMem(Picture[A],Szs);
      BlockRead(Fil,Picture[A]^,Szs);
      Inc(A);
    Until EOF(Atr);
    Close(Fil);
    Close(Atr);
```

Baca file sprite group pertama ke dalam array Picture.

```
    Assign(Fil,'PICS\SPRITE2.PIC');
    Reset(Fil,1);
    Assign(Atr,'PICS\SPRITE2.SZS');
    Reset(Atr);
    Repeat
      Readln(Atr,Szs);
      GetMem(Picture[A],Szs);
      BlockRead(Fil,Picture[A]^,Szs);
```

```
      Inc(A);
    Until EOF(Atr);
    Close(Fil);
    Close(Atr);
```

Baca file sprite group kedua ke dalam array Picture.

```
    For A:=1 to 13 do
      Begin
        Rll_UnCompress(Picture[Object_Char[A].Picture]^,UnComp^);

Get_Dimension(UnComp^,Object_Char[A].Length,Object_Char[A].Width);
      End;
  End;
```

Isikan data panjang dan lebar gambar yang mewakili tiap objek ke dalam variabel Objek_Char.

```
Procedure Put_Display;
Var A:Integer;
  Begin
    PutImage(Process^,5,172,ToolBar^);
```

Tampilkan toolbar pada bawah layar.

```
    If IconTagged<>0 then
      Begin
```

Jika ada icon yang dipilih maka ...

```
        If IconTagged<20 then
          Rectangle(Screen^,5+(IconTagged-1)*21,172,;
                    3+IconTagged*21,187,15) else
        If IconTagged<40 then
          Rectangle(Screen^,5+(IconTagged-21)*16,188,;
                    3+(IconTagged-20)*16,199,15) else
        If IconTagged<60 then
          Rectangle(Screen^,213+(IconTagged-41)*25,188,;
                    211+(IconTagged-40)*25,199,15);
```

Tampilkan persegi panjang yang mengelilingi icon sebagai tanda bahwa icon tersebut telah dipilih.

```
      End;
    For A:=1 to 200 do
      With Enemy_Spot[A] do
        If Tipe<>0 then
          Begin
            Put(Process^,X-Xp,Y-Yp,Object_Char[Tipe].Picture);
          End;
```

Tampilkan seluruh objek pada layar monitor. Gambar dari objek ini dapat diperoleh dari array index Object_Char[Tipe Object].Picture.

Adapun tipe objek ini ialah Tipe pada field dari record Enemy_Spot.

```
    For A:=1 to 1000 do
      With Block[A] do
        If Pic<>0 then
          Begin
            Put(Process^,X-Xp,Y-Yp,Pic);
          End;
```

Tampilkan seluruh balok pada layar monitor.

```
  End;

Function Search_Object(Var Tip:Integer;Xm,Ym:Integer):Integer;
Var A:Integer;
  Begin
```

Procedure ini dipanggil jika kita menekan tombol mouse pada background page. Tujuannya ialah untuk mencari objek yang ada pada koordinat mouse pada saat itu.

```
    Search_Object:=0;
    For A:=1 to 200 do
      With Enemy_Spot[A] do
        If Tipe<>0 then
          Begin
            If (Xm+Xp>X) and (Ym+Yp>Y) and;
                (Xm+Xp<X+Object_Char[Tipe].Length) and
            (Ym+Yp<Y+Object_Char[Tipe].Width) then
              Begin
```

Jika objek tersebut berada pada koordinat mouse maka return function ialah nomor dari objek tersebut.

Function ini digunakan pada saat Anda menggunakan fasilitas DEL dan MOV.

```
                Tip:=Tipe;
                Search_Object:=A;
```

Tip adalah tipe dari objek yang ditemukan sedangkan return function ialah nomor dari objek tersebut.

```
                A:=200;
              End;
          End;
  End;

Function Search_Block(Var Tip:Integer;Xm,Ym:Integer):Integer;
```

```
Var A:Integer;
  Begin
```

Function ini berguna untuk mengetahui balok yang berada pada koordinat mouse sekarang ini. Tujuannya sih sama persis dengan function Search_Object di atas.

Digunakan pada saat fasilitas MOV atau DEL dipilih.

```
    Search_Block:=0;
    For A:=1 to 1000 do
      With Block[A] do
        If Pic<>0 then
          Begin
            If (Xm+Xp>X) and (Ym+Yp>Y) and (Xm+Xp<X+16) and
            (Ym+Yp<Y+11) then
              Begin
```

Jika balok tersebut berada pada koordinat mouse maka ...

```
                Case Pic of
                  P_Brown:Tip:=21;
                  M_Red :Tip:=22;
                  M_Green:Tip:=23;
                  M_Blue:Tip:=24;
                  H_Red :Tip:=25;
                  H_Brown:Tip:=26;
                  H_Gray:Tip:=27;
                  Mold1 :Tip:=28;
                  P_Red :Tip:=29;
                  E_Box :Tip:=30;
                  P_Green:Tip:=31;
                  P_Gray:Tip:=32;
                  Mold2 :Tip:=33;
                End;
```

Return dari function ialah nomor dari balok yang ditemukan. Tip adalah tipe dari balok yang ditemukan.

```
                Search_Block:=A;
                A:=1000;
              End;
          End;
  End;

Procedure Move_Object;
Var Tipe,Obj:Integer;
  Begin
```

Procedure ini digunakan untuk memindahkan sebuah objek/balok ke tempat lain.

```
Obj:=Search_Object(Tipe,Xm,Ym);
```

Mula-mula cari objek yang berada pada posisi mouse.

```
If Obj=0 then
  Begin
    Obj:=Search_Block(Tipe,Xm,Ym);
```

Jika objek tidak ditemukan maka cari balok yang berada pada posisi mouse.

```
    If Obj<>0 then
      Begin
        Block[Obj].Pic:=0;
        IconTagged:=Tipe;
      End;
```

Jika ditemukan balok yang berada pada posisi mouse maka hapus balok yang ditemukan dan set IconTagged = Tipe sehingga kita sekarang diharuskan untuk meletakkan balok yang bertipe sama dengan yang telah kita hapus pada suatu tempat. Dengan cara inilah teknik pemindahan balok dilakukan.

```
    End Else
    Begin
      IconTagged:=Tipe;
      Enemy_Spot[Obj].Tipe:=0;
```

Jika ditemukan objek pada posisi mouse maka hapus objek yang ditemukan dan set IconTagged = Tipe dari objek yang ditemukan sehingga kita diharuskan untuk meletakkan objek tersebut pada layar.

```
    End;
  End;

Procedure Delete_Object;
Var Tipe,Obj:Integer;
  Begin
```

Procedure ini bertujuan untuk menghapus objek/balok pada layar monitor. Adapun teknik dari procedure ini hampir sama seperti procedure Move_Object.

```
  Obj:=Search_Object(Tipe,Xm,Ym);
  if Obj=0 then
    Begin
      Obj:=Search_Block(Tipe,Xm,Ym);
      If Obj<>0 then
        Block[Obj].Pic:=0;
```

IconTagged tidak perlu diset = tipe karena kita hanya bertujuan untuk menghapus dan tidak perlu meletakkan balok berjenis sama pada layar.

```
      End Else
      Begin
        Enemy_Spot[Obj].Tipe:=0;
```

IconTagged tidak perlu diset = tipe karena kita hanya bertujuan untuk menghapus dan tidak perlu meletakkan objek berjenis sama pada layar.

```
      End;
  End;

Function Get_Object_Tipe(No:Integer):Integer;
  Begin
    Case No of
      1..13:Get_Object_Tipe:=No;
    End;
```

Ambil tipe dari objek yang dikirimkan lewat parameter.

```
  End;

Procedure Save_Mission;
  Begin
    Assign(Mission_File,'Pics\Mission.001');
    Rewrite(Mission_File,1);
    BlockWrite(Mission_File,Object_Char,20*SizeOf(Char_Rec));
    BlockWrite(Mission_File,Enemy_Spot,200*SizeOf(Enemy_Rec));
    BlockWrite(Mission_File,Block,1000*SizeOf(Block_Rec));
    Close(Mission_File);
```

Simpan data level pada file MISSION.001. Adapun data yang disimpan ialah data gambar objek, data balok dan data objek.

```
  End;

Procedure Put_Center(Y:Integer;Col1,Col2:Byte;St:String);
  Begin
    PutString(Screen^,150-(Length(St) div 2)*6,Y,Col1,Col2,St);
  End;
```

Procedure di atas ialah procedure untuk menampilkan tulisan pada tengah kolom.

```
Procedure Show_Credit;
  Begin
    VgaCad.Block(Screen^,40,40,270,130,8);
    Rectangle(Screen^,40,40,270,130,7);
    Put_Center(50,0,15,'BUG HUNTER MISSION EDITOR');
    Put_Center(70,0,14,'CREATED BY AGUSTINUS NALWAN');
    Put_Center(90,0,11,'APROMPT ENTERTAINMENT DIVISION');
```

```
    Put_Center(110,0,12,'COPYRIGHT SEPTEMBER 28 1995');
    Ch:=Readkey;
  End;
```

Procedure di atas digunakan untuk menampilkan kredit dari EDITOR ini.

```
Procedure Detect_Mouse_Command;
Var Opt:Integer;
  Begin
    If (Bt=1) and Not Same then
```

Jika tombol mouse yang ditekan sekarang lain dengan yang lalu, artinya bukan merupakan sisa tombol yang ditahan.

```
      Begin
        If Ym>=172 then
          Begin
            If Ym<188 then
              Begin
                If Xm<278 then
                  Begin
                    Opt:=((Xm-5) div 21) + 1;
                    IconTagged:=Opt;
```

Group Icon yang dipilih ialah Group Icon object.

Lakukan perhitungan koordinat mouse sehingga diperoleh icon object mana yang ditekan.

```
                  End;
              End Else
              Begin
                If Xm<213 then
                  Begin
                    Opt:=((Xm-5) div 16) + 21;
                    IconTagged:=Opt;
```

Group Icon yang dipilih ialah Group Icon balok.

Lakukan perhitungan koordinat mouse sehingga diperoleh icon balok mana yang ditekan.

```
                  End Else
                  Begin
                    Opt:=((Xm-209) div 25) + 41;
                    IconTagged:=Opt;
```

Group Icon yang dipilih ialah Group Icon Command (DEL, MOV, SAVE, QUIT).

Lakukan perhitungan untuk menentukan icon command yang ditekan.

```
        End;
    End;
  If (Xm>278) and (Xm<313) and (Ym<189) then
    Begin
      Show_Credit;
      IconTagged:=0;
```

Jika icon credit yang ditekan maka tampilkan kredit.

```
    End;
End Else
Begin
```

Jika ternyata daerah yang ditekan adalah daerah layar maka ...

```
  If (IconTagged>20) and (IconTagged<41) then
    Begin
```

Jika icon yang sedang terpilih ialah salah satu dari icon balok maka ...

```
      For A:=1 to 1000 do
        With Block[A] do
          If Pic=0 then
            Begin
              Pic:=Get_Picture(IconTagged);
              X:=Xm+Xp;Y:=Ym+Yp;
              A:=1000;
```

Letakkan balok pada posisi layar tersebut dengan cara memasukkannya ke dalam daftar balok yang aktif yaitu pada array Block.

```
            End;
    End Else
  If (IconTagged<>0) and (IconTagged<41) then
    Begin
```

Jika yang aktif ialah icon objek maka ...

```
      For A:=1 to 200 do
        With Enemy_Spot[A] do
          If Tipe = 0 then
            Begin
              X:=Xm+Xp;Y:=Ym+Yp;
              Tipe:=Get_Object_Tipe(IconTagged);
              A:=200;
```

Letakkan objek pada posisi layar tersebut dengan cara memasukkannya ke dalam array Enemy_Spot.

```
            End;
    End Else
    Case IconTagged of
```

```
            42:Move_Object;
            43:Delete_Object;
          End;
```

Jika yang dipilih ialah icon command maka lakukan perintah sesuai dengan icon yang dipilih.

```
        End;
      Case IconTagged of
        41:Save_Mission;
        44:Begin
             Sound(100);Delay(30);NoSound;
             VgaCad.Block(Screen^,80,70,240,90,8);
             VgaCad.Rectangle(Screen^,80,70,240,90,7);
             Put_Center(80,0,15,'EXIT EDITTING (Y/N)?');
             Ch:=Readkey;
             If (Ch='y') or (Ch='Y') then Finish:=True;
           End;
      End
```

Jika icon yang dipilih ialah SAVE maka lakukan proses saving mission.
Jika icon yang dipilih ialah QUIT maka pastikan bahwa user akan keluar. Jika ya maka set FINISH = TRUE untuk keluar dari EDITOR.

```
    End;
  End;

Procedure Main;
  Begin
    Finish:=False;
    Xp:=0;Yp:=0;
    ScSpeed:=0;
```

Set kecepatan awal momentum = 0, koordinat parallax scroll = (0, 0).

```
    Repeat
      Repeat
        Put_Parallax_Scroll(Xp,Yp,Process^,UnComp^);
```

Letakkan parallax scroll pada virtual screen.

```
        Put_Display;
```

Tampilkan semua sprite beserta toolbar.

```
        ReadMouse;
```

Baca koordinat mouse yang baru.

```
        Detect_Mouse_Command;
```

Lakukan perintah yang diinstruksikan lewat mouse.

```
If IconTagged=0 then
  PutMouse(Process^,2) else
  PutMouse(Process^,4);
```

Jika IconTagged=0 maka tidak ada icon yang dipilih, jadi panggil PUT-MOUSE dengan parameter 2 yang akan menampilkan kursor mouse sebagai tanda +. Jika tidak maka tampilkan kursor mouse dengan gambar yang berkenaan dengan icon yang dipilih.

```
MoveTo(Process^,Screen^);
If (Xm=319) or (Xm=0) or (Ym=199) or (Ym=0) then
  Inc(ScSpeed)
```

Jika kursor mouse mencapai batas layar maka tambah kecepatan momentum.

```
Else
  ScSpeed:=0;
```

Jika tidak set kecepatan momentum menjadi 0 sehingga layar tidak akan bergeser.

```
If ScSpeed>50 then ScSpeed:=50;
```

Batasi pada terminal velocity = 50.

```
If Xm=319 then Inc(Xp,ScSpeed);
If Xm=0 then Dec(Xp,ScSpeed);
If Ym=199 then Inc(Yp,ScSpeed);
If Ym=0 then Dec(Yp,ScSpeed);
```

Deteksi batas layar yang mana yang ditabrak oleh kursor mouse sehingga program dapat melakukan proses scroll ke arah yang diinginkan.

```
Case Ch of
  #72:Dec(Yp,1);
  #80:Inc(Yp,1);
  #75:Dec(Xp,1);
  #77:Inc(Xp,1);
End;
```

Deteksi tombol keyboard sebagai alat scroll sekunder.

```
If Xp<0 then Xp:=0;
If Yp<0 then Yp:=0;
If Xp>2680 then Xp:=2680;
If Yp>300 then Yp:=300;
```

Batasi scroll dari parallax scrolling.

```
Until Keypressed or Finish;
```

```
      If Not Finish then Ch:=Readkey;
    Until Finish;
  End;

Procedure Init_Mission;
  Begin
    Assign(Mission_File,'Pics\Mission.001');
    Reset(Mission_File,1);
    BlockRead(Mission_File,Object_Char,20*SizeOf(Char_Rec));
    BlockRead(Mission_File,Enemy_Spot,200*SizeOf(Enemy_Rec));
    BlockRead(Mission_File,Block,1000*SizeOf(Block_Rec));
    Close(Mission_File);
```

Pada awal dari program, EDITOR selalu akan membaca file MISSION.001 dengan asumsi file ini telah ada pada disk Anda. Dengan cara ini maka Anda tidak perlu melakukan proses loading setiap kali ingin melanjutkan mengedit level yang telah Anda buat sebelumnya.

```
  End;


Procedure Removal;
  Begin
    CloseVga256;
  End;

Begin
  Preparation;
  Init_Picture;
  Init_Mission;
  Main;
  Removal;
End.
```

Pembuatan EDITOR ini tidak terlalu sulit kan? Nah sekarang kita akan mulai membahas tentang game engine dari BUG HUNTER.

PEMBUATAN GAME ENGINE

Struktur data dari game engine ini hampir sama dengan struktur data yang kita gunakan pada BUG HUNTER MISSION EDITOR di atas. Bedanya di sini kita menambahkan beberapa array baru untuk mengakses animasi, peluru, text dan juga dua buah array baru yang berfungsi sebagai buffer untuk balok dan objek yang akan diakses (mengingat hanya objek dan balok yang dekat dengan Buggy saja yang akan diakses).

Baiklah kita mulai saja sekarang. Lihatlah listing program dari BUG HUNTER di bawah ini.

```
{-----------------------------------------}
{ Program: B_ENG15.PAS                    }
{ Coder  : Agustinus Nalwan               }
{ Fungsi : Game Bug Hunter.               }
{-----------------------------------------}

Uses Crt,VgaCad,Dos,VgaFont;
Const Spot     =  1;
      Shower   = 22;
      Piranha  = 20;
      Ghost    = 28;
      Cloud    = 33;
      Peer     = 36;
      Wheel    = 40;
      Snake    = 42;
      Teeth    = 44;

      P_Brown  = 47;
      P_Red    = 55;
      P_Green  = 57;
      P_Gray   = 58;
      M_Red    = 48;
      M_Green  = 49;
      M_Blue   = 50;
      H_Red    = 51;
      H_Brown  = 52;
      H_Gray   = 53;
      E_Box    = 56;
      Mold1    = 54;
      Mold2    = 59;

      Bomb     = 88;
      Misile   = 91;
      Coin     = 92;
      Invul    = 95;

Type Enemy_Rec = Record
       X,Y    : Integer;
       Tipe   : Byte;
       Energy : Integer;
     End;

     Block_Rec = Record
       X,Y    : Integer;
       Pic    : Byte;
     End;

     Char_Rec = Record
       Length,Width: Word;
       Picture     : Integer;
     End;
```

```
Obj_Rec = Record
  Obj_Tagged: Integer;
  X,Y       : Integer;
  Vx,Vy     : Integer;
  Ctr1,Ctr2,
  Ctr3,Ctr4,Ctr5
            : Integer;
  Direction : Byte;
End;

Fire_Rec = Record
  X, Y      : Integer;
  Vx,Vy     : Integer;
  Pic       : Integer;
  Ctr1,Ctr2,Ctr3,
  Ctr4,Ctr5 : Integer;
  Direction : Byte;
  Destruct  : Byte;
  Tipe      : Byte;
End;

Player_Rec = Record
  Active    : Boolean;
  X,Y       : Integer;
  Vx,Vy     : Integer;
  Direction : Byte;
  Status    : Byte;
  Life      : Byte;
  FCtr      : Integer;
  Ctr1,Ctr2,Ctr3,
  Ctr4,Ctr5 : Integer;
  Score     : Integer;
  PowerUp   : Boolean;
  Misile    : Integer;
  InVul     : Boolean;
  InCtr     : Integer;
End;

P_Rec = Record
  Length,Width:Word;
End;

Mess_Rec = Record
  St:String;
  X,Y:Integer;
  Active:Boolean;
  Col:Byte;
  Ctr:Byte;
End;

Anim_Rec = Record
  X,Y  : Integer;
  Ctr  : Integer;
  Tipe : Byte;
```

```
      Active: Boolean;
    End;

Var Fil         : File;
    Atr         : Text;
    A           : Integer;
    Picture     : Array[1..200] of Pointer;
    PicSize     : Array[1..200] of P_Rec;
    Finish      : Boolean;
    Ch          : Char;
    Szs         : Word;
    Object_Char: Array[1..20] of Char_Rec;
    Enemy_Spot : Array[1..200] of  Enemy_Rec;
    Anim_Act    : Array[1..100] of Anim_Rec;
    Block       : Array[1..1000] of Block_Rec;
    Object_Act : Array[1..40] of Obj_Rec;
    Block_Act   : Array[1..200] of Integer;
    Fire_Act    : Array[1..100] of Fire_Rec;
    Message     : Array[1..20] of Mess_Rec;
    CountDown   : Boolean;
    CtrCD       : Integer;
    My_Player   : Player_Rec;
    Mission_File: File;
    Process     : Pointer;
    UnComp      : Pointer;
    Xp,Yp       : Integer;
    OldInt9     : Pointer;
    PalTable    : Pointer;
    Key         : Byte;
    Flag,Flag1,Flag2,Flag3,Flag4,Flag5,Flag6:Boolean;

procedure int9; interrupt; assembler;

label
  quit, release, tes2, tes3, tes5, testd, testa, testb, testc, ;
  tes4, tes6, teste;
asm
  sti
  in    al,60h
  mov   key,al
  test  al,80h
  jnz   release
  cmp   al,75
  jne   tes2
  mov   flag,1
  jmp   quit

tes2:
  cmp   al,77
  jne   tes3
  mov   flag2,1
  jmp   quit
```

```
tes3:
  cmp   al,72
  jne   tes4
  mov   flag3,1
  jmp   quit

tes4:
  cmp   al,80
  jne   tes5
  mov   flag4,1
  jmp   quit

tes5:
  cmp   al,28
  jne   tes6
  mov   flag5,1
  jmp   quit

tes6:
  cmp   al,57
  jne   quit
  mov   flag6,1
  jmp   quit

release:
  cmp   al,203
  jne   testa
  mov   flag,0
  jmp  quit

testa:
  cmp   al,205
  jne   testb
  mov   flag2,0
  jmp  quit

testb:
  cmp   al,200
  jne   testc
  mov   flag3,0
  jmp  quit

testc:
  cmp   al,208
  jne   testd
  mov   flag4,0
  jmp   quit

testd:
  cmp   al,156
  jne   teste
  mov   flag5,0
  jmp   quit
```

```
teste:
  cmp   al,185
  jne   quit
  mov   flag6,0

quit:
  mov   al,20h
  out   20h,al
end;

Function Strig(Nb:Longint): String;
Var S:String[10];
  Begin
    Str(Nb,S);
    Strig:=S;
  End;
Procedure Small;External;
{$l Small.Obj}

Procedure Get_Dimension(Var Pic;Var X,Y:Word);
Var Length,Width:Word;
  Begin
    Asm
      Push ds
      Les di, Pic
      Mov Ax,[ES:DI]
      Mov Length,Ax
      Add di,2
      Mov Ax,[ES:DI]
      Mov Width,Ax
      Pop ds
    End;
    X:=Length;
    Y:=Width;
  End;

Function Collision(No:Byte;X,Y:Integer;Xl,Yl:Integer):Boolean;
Var A:Integer;
  Begin
    A:=1;
    Collision:=False;
    If (Y+YL>462) then Collision:=True else
    Repeat
      If (X+XL>Block[Block_Act[A]].X+2) and;
         (X<Block[Block_Act[A]].X+12) and
        (Y+YL>Block[Block_Act[A]].Y) and;
         (Y<Block[Block_Act[A]].Y+9) then
          Begin
            Collision:=True;
            A:=200;
          End;
      Inc(A);
    Until (Block_Act[A]=0) or (A=201);
  End;
```

```
Procedure Make_Anim(No:Byte;Xs,Ys:Integer);
Var A:Integer;
  Begin
    For A:=1 to 100 do
      With Anim_Act[A] do
        If not Active then
          Begin
            Tipe:=No;
            Active:=True;
            Ctr:=1;
            X:=Xs;Y:=Ys;
            A:=100;
          End;
  End;

Procedure Create_Message(Sts:String;Xs,Ys:Integer;Cols:Byte);
Var A:Integer;
  Begin
    For A:=1 to 20 do
      If Message[A].Active=False then
        With Message[A] do
          Begin
            X:=Xs;Y:=Ys;
            St:=Sts;Col:=Cols;
            Active:=True;
            Ctr:=1;
            A:=20;
          End;
  End;

Procedure Put(Var Proc;X,Y:Integer;Pic:Integer);
Var Xl,Yl:Word;
  Begin
    If (X<319) and (Y<199) and (X>-50) and (Y>-50) then
      Begin
        If (X+PicSize[Pic].Length>0) and ;
           (Y+PicSize[Pic].Width>0) then
          Begin
            Rll_UnCompress(Picture[Pic]^,UnComp^);
            PutImage(Proc,X,Y,UnComp^);
          End;
      End;
  End;

Procedure PutTrans(Var Proc;X,Y:Integer;Pic:Integer);
Var Xl,Yl:Word;
  Begin
    If (X<319) and (Y<199) and (X>-50) and (Y>-50) then
      Begin
        If (X+PicSize[Pic].Length>0) and ;
           (Y+PicSize[Pic].Width>0) then
          Begin
            Rll_UnCompress(Picture[Pic]^,UnComp^);
```

```
            T_PutImage(Proc,X,Y,UnComp^,PalTable^);
          End;
      End;
  End;

Procedure Init_Environment;
  Begin
    OpenVga256;
    LoadFont(@Small);
    Install_Palette('Palette.Pal');
    Load_Parallax_Scroll('Back1');
    GetMem(PalTable,65280);
    Install_Transparent(PalTable,'Trans.Tab');
  End;

Procedure Init_Variable;
  Begin
    CtrCD:=30;
    CountDown:=False;
    Xp:=0;Yp:=300;
    GetMem(Process,64000);
    GetMem(UnComp,10000);
    FillChar(Block,5000,0);
    FillChar(Enemy_Spot,1400,0);
    FillChar(Object_Act,40*SizeOf(Obj_Rec),0);
    FillChar(Fire_Act,100*SizeOf(Fire_Rec),0);
    FillChar(Block_Act,200*SizeOf(Block_Rec),0);
    FillChar(Anim_Act,100*SizeOf(Anim_Rec),0);
    FillChar(Message,20*SizeOf(Mess_Rec),0);
    With My_Player do
      Begin
        InVul:=False;
        InCtr:=0;
        Misile:=0;
        Life:=6;
        PowerUp:=False;
        Score:=0;
        Active:=True;
        X:=50;
        Y:=400;
        Vx:=0;Vy:=0;
        Ctr1:=0;Ctr2:=0;Ctr3:=1;Ctr4:=0;Ctr5:=1;
        Status:=1;
        Direction:=1;
        FCtr:=1;
      End;
  End;

Procedure Init_Device;
  Begin
    Randomize;
    GetIntVec($9,OldInt9);
    SetIntVec($9,@Int9);
    Flag:=False;
```

```
    Flag1:=False;
    Flag2:=False;
    Flag3:=False;
    Flag4:=False;
    Flag5:=False;
    Flag6:=False;
  End;

Procedure Init_Picture;
  Begin
    Assign(Fil,'PICS\SPRITE.PIC');
    Reset(Fil,1);
    Assign(Atr,'PICS\SPRITE.SZS');
    Reset(Atr);
    Finish:=False;
    A:=1;
    Repeat
      Readln(Atr,Szs);
      GetMem(Picture[A],Szs);
      BlockRead(Fil,Picture[A]^,Szs);
      Inc(A);
    Until EOF(Atr);
    Close(Fil);
    Close(Atr);

    Assign(Fil,'PICS\SPRITE2.PIC');
    Reset(Fil,1);
    Assign(Atr,'PICS\SPRITE2.SZS');
    Reset(Atr);
    Repeat
      Readln(Atr,Szs);
      GetMem(Picture[A],Szs);
      BlockRead(Fil,Picture[A]^,Szs);
      Inc(A);
    Until EOF(Atr);
    Close(Fil);
    Close(Atr);

    Assign(Fil,'PICS\SPRITE3.PIC');
    Reset(Fil,1);
    Assign(Atr,'PICS\SPRITE3.SZS');
    Reset(Atr);
    Repeat
      Readln(Atr,Szs);
      GetMem(Picture[A],Szs);
      BlockRead(Fil,Picture[A]^,Szs);
      Inc(A);
    Until EOF(Atr);
    Close(Fil);
    Close(Atr);

    Assign(Fil,'PICS\SPRITE4.PIC');
    Reset(Fil,1);
    Assign(Atr,'PICS\SPRITE4.SZS');
```

```
    Reset(Atr);
    Repeat
      Readln(Atr,Szs);
      GetMem(Picture[A],Szs);
      BlockRead(Fil,Picture[A]^,Szs);
      Inc(A);
    Until EOF(Atr);
    Close(Fil);
    Close(Atr);
    For B:=1 to A-1 do
      Begin
        Rll_UnCompress(Picture[B]^,UnComp^);
        Get_Dimension(UnComp^,PicSize[B].Length,PicSize[B].Width);
      End;
  End;

Procedure Put_Center(Y:Integer;Col1,Col2:Byte;St:String);
  Begin
    PutString(Screen^,150-(Length(St) div 2)*6,Y,Col1,Col2,St);
  End;

Procedure Show_Credit;
  Begin
    VgaCad.Block(Screen^,40,40,270,130,8);
    Rectangle(Screen^,40,40,270,130,7);
    Put_Center(50,0,15,'BUG HUNTER MISSION EDITOR');
    Put_Center(70,0,14,'CREATED BY AGUSTINUS NALWAN');
    Put_Center(90,0,11,'APROMPT ENTERTAINMENT DIVISION');
    Put_Center(110,0,12,'COPYRIGHT SEPTEMBER 28 1995');
    Ch:=Readkey;
  End;

Procedure Init_Mission;
  Begin
    Assign(Mission_File,'Pics\Mission.001');
    Reset(Mission_File,1);
    BlockRead(Mission_File,Object_Char,20*SizeOf(Char_Rec));
    BlockRead(Mission_File,Enemy_Spot,200*SizeOf(Enemy_Rec));
    BlockRead(Mission_File,Block,1000*SizeOf(Block_Rec));
    Close(Mission_File);
    For A:=1 to 1000 do
      If Enemy_Spot[A].Tipe=1 then
        Begin
          My_Player.X:=Enemy_Spot[A].X;
          My_Player.Y:=Enemy_Spot[A].Y;
          A:=1000;
        End;
  End;

Procedure Detect_Keyboard;
  Begin
  With My_Player do
   If (Status<>4) and (Life>0) then
   Begin
```

```
    If Flag then
      Begin
        Vx:=Vx-1;
        If Status<>3 then Status:=2;
        If Vx<-6 then Vx:=-6;
        Direction:=0;
      End;
    If (Flag2) then
      Begin
        Vx:=Vx+1;
        If Status<>3 then Status:=2;
        If Vx>6 then Vx:=6;
        Direction:=1;
      End;
    If Flag3 then
      Begin
        If (Status=3) and (Ctr1<12) then
          Begin
            Vy:=Vy-1;
            Status:=3;
          End;
        If (Status<>3) then
          Begin
            Vy:=-5;
            Status:=3;
            Ctr1:=0;
          End;
      End;
     If Flag6 and (My_Player.Ctr3=0) and My_Player.PowerUp and;
                  (My_Player.Misile>0) then
       Begin
         Dec(My_Player.Misile);
         For A:=1 to 20 do
           With Fire_Act[A] do
             If Tipe=0 then
               Begin
                 Tipe:=1;
                 X:=My_Player.X+10;
                 Y:=My_Player.Y+4;
                 If My_Player.Direction=1 then Vx:=7 else Vx:=-7;
                 My_Player.Ctr3:=20;
                 My_Player.FCtr:=1;
                 My_Player.Status:=5;
                 Pic:=67;
                 Ctr1:=0;
                 Vy:=0;
                 A:=20;
               End;
       End;
End;
  End;

Procedure Enemy_Startup(A,B:Integer);
  Begin
```

```
With Object_Act[B] do
  Case Enemy_Spot[A].Tipe of
    2:;
    3:Begin
        Vx:=0;
        Vy:=0;
        Ctr2:=0;
        Ctr1:=0;
        Ctr3:=0;
        Ctr5:=0;
        Direction:=1;
      End;
    4:Begin
        Vx:=1;
        Vy:=0;
        Ctr1:=1;
        Ctr2:=1;
        Ctr5:=0;
        Direction:=1;
      End;
    5:Begin
        If My_Player.X<Enemy_Spot[A].X then Vx:=-3 else Vx:=3;
        Vy:=-8;
        If Vx<0 then Direction:=0 else Direction:=1;
        Ctr1:=1;Ctr2:=1;
        Ctr5:=0;
      End;
    6:Begin
        Vx:=-2;
        Vy:=0;
        Ctr1:=1;
        Direction:=1;
        Ctr5:=0;
      End;
    7:Begin
        If My_Player.X<Enemy_Spot[A].X then Vx:=-3 else Vx:=3;
        Vy:=0;
        If Vx<0 then Direction:=0 else Direction:=1;
        Ctr1:=1;
        Ctr2:=0;
        Ctr3:=0;
        Ctr5:=0;
      End;
    8:Begin
        Vx:=0;
        Vy:=0;
        Ctr1:=1;
        Ctr2:=0;
        Ctr3:=0;
        Ctr4:=0;
        Ctr5:=0;
        Direction:=1;
      End;
    10..13:Begin
```

```
            Vx:=0;
            Vy:=0;
            Ctr1:=1;
            Ctr2:=0;
            Ctr3:=0;
            Ctr4:=0;
            Ctr5:=0;
            Direction:=1;
          End;
      End;
  End;

Function GetPicture(No,Ctr,Direction:Integer):Integer;
  Begin
    GetPicture:=0;
    Case No of
      2:Begin;

        End;
      3:Case Ctr of
          1:GetPicture:=27;
          2:GetPicture:=26;
          3:GetPicture:=25;
          4:GetPicture:=24;
          5:GetPicture:=23;
          6:GetPicture:=22;
          7:GetPicture:=23;
          8:GetPicture:=22;
          9:GetPicture:=23;
          10:GetPicture:=22;
          11:GetPicture:=23;
          12:GetPicture:=24;
          13:GetPicture:=25;
          14:GetPicture:=26;
          15:GetPicture:=27;
        End;
      4:Begin
        If Direction=1 then
          Case Ctr of
            1:GetPicture:=20;
            2:GetPicture:=21;
          End Else
          Case Ctr of
            1:GetPicture:=119;
            2:GetPicture:=120;
          End;
        End;
      5:Begin
        Case Ctr of
          1:GetPicture:=39;
          2:GetPicture:=38;
          3:GetPicture:=37;
          4..100:GetPicture:=36;
        End;
```

```
    End;
  6:Begin
      Case Ctr of
        1:GetPicture:=40;
        2:GetPicture:=41;
      End;
    End;
  7:Begin
    If Direction=1 then
      Case Ctr of
        1:GetPicture:=33;
        2:GetPicture:=34;
      End Else
      Case Ctr of
        1:GetPicture:=121;
        2:GetPicture:=122;
      End;
    End;
  8:Begin
    If Direction=0 then
      Case Ctr of
        1:GetPicture:=44;
        2:GetPicture:=45;
        3:GetPicture:=46;
        4:GetPicture:=45;
      End Else
      Case Ctr of
        1:GetPicture:=126;
        2:GetPicture:=127;
        3:GetPicture:=128;
        4:GetPicture:=127;
      End;
    End;
 10:Begin
      Case Ctr of
        1:GetPicture:=86;
        2:GetPicture:=87;
        3:GetPicture:=88;
        4:GetPicture:=87;
      End;
    End;
 11:Begin
      Case Ctr of
        1:GetPicture:=89;
        2:GetPicture:=90;
        3:GetPicture:=91;
        4:GetPicture:=90;
      End;
    End;
 13:Begin
      Case Ctr of
        1:GetPicture:=92;
        2:GetPicture:=93;
        3:GetPicture:=94;
```

```
            4:GetPicture:=93;
          End;
        End;
      12:Begin
          Case Ctr of
            1:GetPicture:=95;
            2:GetPicture:=96;
            3:GetPicture:=97;
            4:GetPicture:=96;
          End;
        End;
    End;
  End;

Procedure Scanning_Area;
Var A,B:Integer;
    Cancel:Boolean;
    St:String;
  Begin
    For A:=1 to 40 do
     IF Object_Act[A].Obj_Tagged<>0 then
      With Enemy_Spot[Object_Act[A].Obj_Tagged] do
        Begin
          If (Object_Act[A].X+Object_Char[Tipe].Length<Xp) or ;
               (Object_Act[A].X>Xp+319) or ;
                 (Object_Act[A].Y+Object_Char[Tipe].Width<Yp)
          or (Object_Act[A].Y>Yp+199) then
               If Tipe >1 then
                 Begin
                   Object_Act[A].Obj_Tagged:=0;
                   Case Tipe of
                     2:;
                     3:Tipe:=Tipe;
                     4:Tipe:=0;
                     5:Tipe:=0;
                     6:Tipe:=0;
                     7:Tipe:=Tipe;
                     8:Tipe:=0;
                     9..12:Tipe:=Tipe;
                   End;
                 End;
        End;
    For A:=1 to 200 do
      With Enemy_Spot[A] do
        Begin
          If (X+Object_Char[Tipe].Length>Xp) and (X<Xp+319) and
             (Y+Object_Char[Tipe].Width>Yp)  and (Y<Yp+199) and ;
              (Tipe >1) then
               Begin
                 Cancel:=False;
                 For B:=1 to 40 do
                   Begin
                     If Object_Act[B].Obj_Tagged=A then
                       Begin
```

```
                        Cancel:=True;
                        B:=40;
                      End;
                  End;
               If Not Cancel then
                For B:=1 to 40 do
                  If Object_Act[B].Obj_Tagged=0 then
                    Begin
                        Object_Act[B].Obj_Tagged:=A;
                        Object_Act[B].X:=X;
                        Object_Act[B].Y:=Y;
                        Enemy_Startup(A,B);
                        B:=40;
                    End;
             End;
        End;
    B:=0;
    For A:=1 to 1000 do
      With Block[A] do
        If Pic<>0 then
          Begin
            If (X<Xp+379) and (X+16>Xp-50) and (Y<Yp+260) and ;
                (Y+11>Yp-60) then
              Begin
                Inc(B);
                Block_Act[B]:=A;
              End;
          End;
Inc(B);
    Block_Act[B+1]:=0;
  End;

Procedure Redraw_Screen;
Var A,Pic:Integer;
  Begin
    For A:=1 to 100 do
      With Fire_Act[A] do
        If Tipe<>0 then
          Begin
            Put(Process^,X-Xp,Y-Yp,Pic);
          End;
    For A:=1 to 20 do
      With Message[A] do
        If Active then
          Begin
            If (Y>Yp+1) and (Y+8<Yp+199) then ;
                PutString(Process^,X-Xp,Y-Yp,1,Col,St);
          End;
    For A:=1 to 200 do
      If Block_Act[A]=0 then
        Begin
          A:=200;
        End Else
        With Block[Block_Act[A]] do
```

```
      Begin
        Put(Process^,X-Xp,Y-Yp,Pic);
      End;
For A:=1 to 100 do
  With Anim_Act[A] do
    If Active then
      Begin
        Case Tipe of
          1:Begin
              Case Ctr of
                1..3:Pic:=70;
                4..6:Pic:=71;
                7..9:Pic:=72;
                10..12:Pic:=73;
              End;
            End;
          2:Begin
              Case Ctr of
                1..3:Pic:=79;
                4..6:Pic:=80;
              End;
            End;
          3:Begin
              Case Ctr of
                1..3:Pic:=74;
                4..6:Pic:=75;
                7..9:Pic:=76;
                10..12:Pic:=77;
              End;
            End;
          4:Begin
              Case Ctr of
                1..3:Pic:=61;
                4..6:Pic:=62;
                7..9:Pic:=63;
                10..12:Pic:=61;
              End;
            End;
          5:Begin
              Case Ctr of
                1..3:Pic:=85;
                4..6:Pic:=84;
                7..9:Pic:=83;
                10..12:Pic:=82;
                13..15:Pic:=81;
              End;
            End;
          6:Begin
              Case Ctr of
                1..2:Pic:=64;
                3..4:Pic:=65;
                5..6:Pic:=66;
                7..8:Pic:=65;
              End;
```

```
                X:=My_Player.X-10;Y:=My_Player.Y-10;
              End;
          End;
          If Tipe in[5,6] then PutTrans(Process^,X-Xp,Y-Yp,Pic);
              else
            Put(Process^,X-Xp,Y-Yp,Pic);
        End;
    For A:=1 to 40 do
      If Object_Act[A].Obj_Tagged<>0 then
        With Object_Act[A] do
          Begin

Pic:=GetPicture(Enemy_Spot[Obj_Tagged].Tipe,Ctr1,Direction);
          If Pic<>0 then
            If Ctr5=0 then Put(Process^,X-Xp,Y-Yp,Pic) else
            PutTrans(Process^,X-Xp,Y-Yp,Pic);
        End;
    With My_Player do
     Begin
     If PowerUp then
      Begin
      If Direction=1 then
        Begin
          Case Status of
            1:Pic:=12;
            2:Case FCtr of
                1..3:Pic:=11;
                4..6:Pic:=12;
                7..9:Pic:=13;
                10..12:Pic:=14;
                13..15:Pic:=13;
                16..18:Pic:=12;
              End;
            3:Case FCtr of
                1:Pic:=5;
                2:Pic:=6;
                3:Pic:=7;
                4:Pic:=8;
                5:Pic:=9;
                6..200:Begin
                         If FCtr Mod 2 = 1 then Pic:=9 else Pic:=10;
                       End;
              End;
            4:Case FCtr of
                1..100:Pic:=18 + FCtr Mod 2;
              End;
            5:Case FCtr of
                1:Pic:=15;
                2:Pic:=15;
                3:Pic:=16;
                4:Pic:=16;
              End;
          End;
        End Else
```

```
  Begin
    Case Status of
      1:Pic:=109;
      2:Case FCtr of
          1..3:Pic:=108;
          4..6:Pic:=109;
          7..9:Pic:=110;
          10..12:Pic:=111;
          13..15:Pic:=110;
          16..18:Pic:=109;
        End;
      3:Case FCtr of
          1:Pic:=102;
          2:Pic:=103;
          3:Pic:=104;
          4:Pic:=105;
          5:Pic:=106;
          6..200:Begin
                   If FCtr Mod 2 = 1 then Pic:=107 else;
                      Pic:=106;
                 End;
        End;
      4:Case FCtr of
          1..100:Pic:=18 + FCtr Mod 2;
        End;
      5:Case FCtr of
          1:Pic:=112;
          2:Pic:=112;
          3:Pic:=113;
          4:Pic:=113;
        End;
    End;
   End;
End Else
Begin
If Direction=1 then
  Begin
    Case Status of
      1:Pic:=1;
      2:Case FCtr of
          1..3:Pic:=2;
          4..6:Pic:=3;
          7..9:Pic:=4;
          10..12:Pic:=3;
          13..15:Pic:=2;
          16..18:Pic:=1;
        End;
      3:Case FCtr of
          1:Pic:=5;
          2:Pic:=6;
          3:Pic:=7;
          4:Pic:=8;
          5:Pic:=9;
          6..200:Begin
```

```
                    If FCtr Mod 2 = 1 then Pic:=9 else Pic:=10;
                  End;
            End;
          4:Case FCtr of
              1..100:Pic:=18 + FCtr Mod 2;
            End;
        End;
      End Else
      Begin
        Case Status of
          1:Pic:=98;
          2:Case FCtr of
              1..3:Pic:=99;
              4..6:Pic:=100;
              7..9:Pic:=101;
              10..12:Pic:=100;
              13..15:Pic:=99;
              16..18:Pic:=98;
            End;
          3:Case FCtr of
              1:Pic:=102;
              2:Pic:=103;
              3:Pic:=104;
              4:Pic:=105;
              5:Pic:=106;
              6..200:Begin
                       If FCtr Mod 2 = 1 then Pic:=107 else;
                         Pic:=106;
                     End;
            End;
          4:Case FCtr of
              1..100:Pic:=18 + FCtr Mod 2;
            End;
        End;
       End;
      End;
      If Active then
        If Ctr5>30 then Put(Process^,X-Xp,Y-Yp,Pic) else
        PutTrans(Process^,X-Xp,Y-Yp,Pic);
    End;
  PutString(Process^,230,2,0,15,'SCORE
'+Strig(My_Player.Score));
  PutString(Process^,130,2,0,LightGreen,'MISSILE
'+Strig(My_Player.Misile));
  Rll_UnCompress(Picture[1]^,UnComp^);
  For A:=1 to My_Player.Life do
    Begin
      T_Put3D(Process^,5+(A-1)*20,2,16,16,UnComp^,PalTable^);
    End;
End;


Function Check_Move(No:Byte;Xr,Yr:Integer;
                    Var Vx,Vy,Rx,Ry:Integer;Xl,Yl:Integer):Byte;
```

```
Var A:Integer;
    Vrx,Vry:Real;
    Finish:Boolean;
    X,Y:Real;
    Loop,Count:Integer;
  Begin
    X:=Xr;Y:=Yr;
    If Abs(Vx)>Abs(Vy) then
      Begin
        If Vx>0 then Vrx:=1 else Vrx:=-1;
        Loop:=Abs(Vx);
        Vry:=Vy/Abs(Vx);
      End Else
    If Vy<>0 then
      Begin
        If Vy>0 then Vry:=1 else Vry:=-1;
        Vrx:=Vx/Abs(Vy);
        Loop:=Abs(Vy);
      End Else
      Begin
        Vry:=0;Vrx:=0;
        Loop:=1;
      End;
    Finish:=False;
    Count:=0;
    Check_Move:=0;
    Repeat
      Inc(Count);
      If Collision(1,Round(X+Vrx),Round(Vry+Y),Xl,Yl)=False then
        Begin
          X:=X+Vrx;
          Y:=Y+Vry;
        End Else
        Begin
          If Collision(1,Round(Vrx+X),Round(Y),Xl,Yl)=False then
            Begin
              { Height Collision }
              Finish:=True;
              If Vy>=0 then Check_Move:=2 else Check_Move:=4;
            End Else
          If Collision(1,Round(X),Round(Vry+Y),Xl,Yl)=False then
            Begin
              { Side Collision }
              Finish:=True;
              Check_Move:=1
            End Else
            Begin
              { Both Collision }
              Finish:=True;
              Check_Move:=3;
            End;
        End;
      If Count>=Loop then Finish:=True;
    Until Finish;
```

```
    Rx:=Round(X)-Xr;
    Ry:=Round(Y)-Yr;
  End;

Procedure Do_Physics;
Var Rx,Ry : Integer;
    A     : Integer;
    XL,YL : Integer;
    Tipe  : Byte;
  Begin
    For A:=1 to 100 do
      With Fire_Act[A] do
        If (Fire_Act[A].Tipe<>0) and (Fire_Act[A].Tipe<6) then
          Begin
            X:=X+Vx;
            Y:=Y+Vy;
            If (X>Xp+320) or (X<Xp) or (Y>Yp+200) or (Y<Yp) then
              Begin
                Fire_Act[A].Tipe:=0;
              End;
            If Collision(1,X,Y,PicSize[Pic].Length,;
                         PicSize[Pic].Width) then
              Begin
                Make_Anim(Tipe,X,Y);
                Fire_Act[A].Tipe:=0;
              End;
          End;
    With My_Player do
      If Active then
        Begin
          If Collision(1,X,Y+1,PicSize[1].Length,;
                       PicSize[1].Width)=False then
            Begin
              If Vy<9 then Vy:=Vy+1;
            End;
          Tipe:=Check_Move(1,X,Y,Vx,Vy,Rx,Ry,PicSize[1].Length,;
                           PicSize[1].Width);
          Case Tipe of
            0:Begin
                { No Collision }
                If Collision(1,X+Rx,Y+Ry,PicSize[1].Length,;
                             PicSize[1].Width) = False then
                  Begin
                    X:=X+Rx;Y:=Y+Ry;
                  End;
              End;
            1:Begin
                { Side Collision }
                Vx:=0;
                If Collision(1,X+Vx,Y+Vy,PicSize[1].Length,;
                             PicSize[1].Width) = False then
                  Begin
                    X:=X+Vx;Y:=Y+Vy;
                  End;
```

```
          End;
        4:Begin
            { Upper Collision }
            If Collision(1,X+Rx,Y+Ry,PicSize[1].Length,;
                         PicSize[1].Width) = False then
              Begin
                Y:=Y+Ry;
                X:=X+Rx;
              End;
            Vy:=-Vy;
          End;
        2:Begin
            { Bottom Collision }
            Status:=1;
            FCtr:=1;
            Vy:=0;
            If Collision(1,X+Rx,Y+Ry,PicSize[1].Length,;
                         PicSize[1].Width) = False then
              Begin
                X:=X+Rx;Y:=Y+Ry;
              End;
          End;
        3:Begin;
            If Vy>=0 then
              Begin
                Status:=1;
                FCtr:=1;
              End;
            If Collision(1,X+Rx,Y+Ry,PicSize[1].Length,;
                         PicSize[1].Width) = False then
              Begin
                X:=X+Rx;Y:=Y+Ry;
              End;
            Vy:=0;Vx:=0;
          End;
      End;
    End;
For A:=1 to 40 do
With Object_Act[A] do
If Ctr5=0 then
Begin
  Xl:=Object_Char[Enemy_Spot[Obj_Tagged].Tipe].Length;
  Yl:=Object_Char[Enemy_Spot[Obj_Tagged].Tipe].Width;
  If Obj_Tagged<>0 then
    Begin
      If Collision(1,X,Y+1,XL,YL)=False then
        Begin
          Case Enemy_Spot[Obj_Tagged].Tipe of
            1:;
            2:;
            3:;
            4:Begin
                Vx:=-Vx;
                If Direction=1 then Direction:=0 else ;
```

```
                Direction:=1;
              If Collision(1,X+Vx,Y+1,XL,YL)=False then
                If Vy<9 then Vy:=Vy+1;
            End;
          5:Begin
              If Vy<9 then Vy:=Vy+1;
            End;
          6:Begin
              If Vy<9 then Vy:=Vy+1;
            End;
          7:;
          8:Begin
              If Vy<9 then Vy:=Vy+1;
            End;
      End;
    End;
Tipe:=Check_Move(1,X,Y,Vx,Vy,Rx,Ry,XL,YL);
Case Tipe of
  0:Begin
      { No Collision }
      X:=X+Rx;Y:=Y+Ry;
    End;
  1:Begin
      { Side Collision }
      Vx:=-Vx;
      If Direction=1 then Direction:=0 else Direction:=1;
      X:=X+Vx;Y:=Y+Vy;
    End;
  4:Begin
      { Upper Collision }
      Y:=Y+Ry;
      X:=X+Rx;
      Vy:=-Vy;
    End;
  2:Begin
      { Bottom Collision }
      Vy:=0;
      X:=X+Rx;Y:=Y+Ry;
      Case Enemy_Spot[Obj_Tagged].Tipe of
        5:Begin
            Vy:=-8;
            Ctr1:=1;
          End;
        8:Begin
            Ctr2:=1;
            Ctr3:=0;
            Vx:=0;
          End;
      End;
    End;
  3:Begin;
      Vy:=0;Vx:=0;
    End;
End;
```

```
        End;
      End Else
      Begin
        Inc(Y,3);
        If Y>Yp+200 then
          Begin
            Enemy_Spot[Obj_Tagged].Tipe:=0;
            Obj_Tagged:=0;
          End;
      End;
  End;

Procedure Object_Collision;
Var A,B:Integer;
    XL,YL,XM,YM:Integer;
  Begin
   For A:=21 to 100 do
     If (Fire_Act[A].Tipe<>0) And (My_Player.Ctr5>30) and;
        (My_Player.InVul=False) then
       Begin
         XL:=Object_Char[1].Length;
         YL:=Object_Char[1].Width;
         XM:=PicSize[Fire_Act[A].Pic].Length;
         YM:=PicSize[Fire_Act[A].Pic].Width;
         If (Fire_Act[A].X+XM>My_Player.X) and;
            (Fire_Act[A].X<My_Player.X+XL) and
           (Fire_Act[A].Y+YM>My_Player.Y) and;
           (Fire_Act[A].Y<My_Player.Y+YL) then
             Begin
                   With My_Player do
                     Begin
                       Status:=4;
                       FCtr:=1;
                       Ctr1:=1;
                       My_Player.Vy:=-6;
                       Ctr5:=1;
                       Create_Message('1 LIVE LOST',My_Player.X-20,;
                              My_Player.Y-20,LightRed);
                       Dec(Life);
                       If Life=0 then
                         Begin
                           CountDown:=True;
                         End;
                     End;
                 Fire_Act[A].Tipe:=0;
                 Make_Anim(1,Fire_Act[A].X,Fire_Act[A].Y);
                 A:=100;
             End;
       End;
   For A:=1 to 20 do
     If Fire_Act[A].Tipe<>0 then
         For B:=1 to 40 do
           With Object_Act[B] do
             If (Obj_Tagged<>0) and (Ctr5=0) and;
```

```
                (Enemy_Spot[Obj_Tagged].Tipe<9) then
              Begin

XL:=Object_Char[Enemy_Spot[Obj_Tagged].Tipe].Length;
               YL:=Object_Char[Enemy_Spot[Obj_Tagged].Tipe].Width;
               XM:=PicSize[Fire_Act[A].Pic].Length;
               YM:=PicSize[Fire_Act[A].Pic].Width;
               If (Fire_Act[A].X+XM>X) and ;
                   (Fire_Act[A].X<X+XL) and
                  (Fire_Act[A].Y+YM>Y) and ;
                   (Fire_Act[A].Y<Y+YL) then
                    Begin
                     If (Enemy_Spot[Obj_Tagged].Tipe=3) and;
                         (Ctr2<30) then else
                     Begin
                      Ctr5:=1;
                      Ctr1:=1;
                      Make_Anim(4,Fire_Act[A].X,Fire_Act[A].Y);
                      Fire_Act[A].Tipe:=0;
                      Create_Message('100 PTS',Fire_Act[A].X-20,;
                             Fire_Act[A].Y-20,15);
                      Inc(My_Player.Score,100);
                      B:=40;
                     End;
                    End;
              End;
   If (My_Player.Ctr5>30) then
    For A:=1 to 40 do
      With Object_Act[A] do
       If (Obj_Tagged<>0) and (Ctr5=0) then
        Begin
          XL:=Object_Char[Enemy_Spot[Obj_Tagged].Tipe].Length;
          YL:=Object_Char[Enemy_Spot[Obj_Tagged].Tipe].Width;
          XM:=Object_Char[1].Length;
          YM:=Object_Char[1].Width;
          If (My_Player.X+XM>X) and (My_Player.X<X+XL) and
             (My_Player.Y+YM>Y) and (My_Player.Y<Y+YL) then
            Begin
            If (Enemy_Spot[Obj_Tagged].Tipe=3) and ;
                (Ctr2<30) then else
            Begin
            Case Enemy_Spot[Obj_Tagged].Tipe of
              1..8:Begin
              If (My_Player.Vy>0) and;
                  (Enemy_Spot[Obj_Tagged].Tipe<>3) and
             (Enemy_Spot[Obj_Tagged].Tipe<>6) or ;
                 My_Player.Invul then
               Begin
                 Ctr5:=1;
                 Ctr1:=1;
                 My_Player.FCtr:=1;
                 My_Player.Vy:=-6;
                 Create_Message('100 PTS',My_Player.X-20,;
                   My_Player.Y-20,15);
```

```
            Inc(My_Player.Score,100);
          End Else
          With My_Player do
            Begin
              Status:=4;
              FCtr:=1;
              Ctr1:=1;
              My_Player.Vy:=-6;
              Ctr5:=1;
              My_Player.PowerUp:=False;
              Create_Message('1 LIVE LOST',My_Player.X-20,;
                    My_Player.Y-20,LightRed);
              Dec(Life);
              If Life=0 then
                Begin
                  Finish:=True;
                End;
            End;
       End;
       10:Begin
          My_Player.PowerUp:=True;
          Create_Message('GOT MISSILE LAUNCHER',;
               My_Player.X-20,My_Player.Y-20,White);
          Inc(My_Player.Misile,10);
          Enemy_Spot[Object_Act[A].Obj_Tagged].Tipe:=0;
          Object_Act[A].Obj_Tagged:=0;
        End;
      11:Begin
          Inc(My_Player.Misile,10);
          Create_Message('GOT 10 ROCKETS',My_Player.X-20,;
               My_Player.Y-20,Yellow);
          Enemy_Spot[Object_Act[A].Obj_Tagged].Tipe:=0;
          Object_Act[A].Obj_Tagged:=0;
        End;
      12:Begin
          My_Player.InVul:=True;
          My_Player.InCtr:=400;
          Make_Anim(6,My_Player.X-10,My_Player.Y-10);
          Create_Message('INVULNARBILITY ON',;
              My_Player.X-20,My_Player.Y-20,Yellow);
          Enemy_Spot[Object_Act[A].Obj_Tagged].Tipe:=0;
          Object_Act[A].Obj_Tagged:=0;
        End;
      13:Begin
          Inc(My_Player.Score,100);
          Create_Message('100 PTS',My_Player.X-20,;
               My_Player.Y-20,Yellow);
          Enemy_Spot[Object_Act[A].Obj_Tagged].Tipe:=0;
          Object_Act[A].Obj_Tagged:=0;
        End;
  End;
  A:=40;
 End;
End;
```

```
        End;
  End;

Procedure Do_Status;
Var A,B:Integer;
    R,Dx,Dy:Real;
  Begin
    For A:=1 to 100 do
      With Anim_Act[A] do
        If Active then
          Begin
            Inc(Ctr);
            Case Tipe of
              1:If Ctr>12 then Active:=False;
              2:If Ctr>6 then Active:=False;
              3:If Ctr>12 then Active:=False;
              4:If Ctr>12 then Active:=False;
              5:If Ctr>15 then Active:=False;
              6:Begin
                  If Ctr>8 then Ctr:=1;
                  If My_Player.Invul=False then Active:=False;
                End;
              Else Active:=False;
            End;
          End;
    For A:=1 to 100 do
      With Fire_Act[A] do
        If Tipe<>0 then
          Begin
            If A<=20 then
              Begin
                If Round(Random(2))=1 then Make_Anim(5,;
                     X+Round(Random(3))-2,
                Y+Round(Random(3))-2);
                        If Round(Random(2))=1 then Make_Anim(5,;
                            X+Round(Random(3))-2,
                Y+Round(Random(3))-2);
              End;
            Inc(Ctr1);
          End;
    For A:=1 to 20 do
      If Message[A].Active then
        With Message[A] do
          Begin
            Dec(Y,2);
            Inc(Ctr);
            If Ctr>15 then Active:=False;
          End;
    With My_Player do
    Begin
      If InVul then
        Begin
          Dec(InCtr);
          If InCtr<1 then
```

```
      Begin
        InVul:=False;
      End;
  End;
Dec(Ctr3);
If Ctr3<1 then Ctr3:=0;
If Ctr5>0 then
  Begin
    Inc(Ctr5);
    If Ctr5>30 then Ctr5:=32;
  End;
If Not Flag and Not Flag2 then
 Begin
   If Vx<0 then Vx:=Vx+1;
   If Vx>0 then Vx:=Vx-1;
 End;
Case Status of
  2:Begin
      Inc(Fctr);
      If Fctr>18 then Fctr:=1;
      If Vx=0 then Status:=1;
    End;
  3:Begin
      Inc(Fctr);
      If Not Flag and Not Flag2 then
        Begin
          If Vx<0 then Vx:=Vx+1;
          If Vx>0 then Vx:=Vx-1;
        End;
      Inc(Ctr1);
      If Ctr1>=20 then Ctr1:=20;
    End;
  5:Begin
      Inc(FCtr);
      If FCtr>4 then
        Begin
          FCtr:=1;
          Status:=1;
        End;
      If Not Flag and Not Flag2 then
        Begin
          If Vx<0 then Vx:=Vx+1;
          If Vx>0 then Vx:=Vx-1;
        End;
    End;
  4:Begin
      Inc(Ctr1);
      If Ctr1>3 then
        Begin
          Inc(Fctr);
          If Fctr>20 then
            Begin
              FCtr:=1;
              Status:=1;
```

```
            End;
          Ctr1:=0;
        End;
    End;
End;
End;
For A:=1 to 40 do
  With Object_Act[A] do
    If Obj_Tagged<>0 then
     With Enemy_Spot[Obj_Tagged] do
      If Ctr5=0 then
      Begin
        Case Enemy_Spot[Obj_Tagged].Tipe of
          2:;
          3:Begin
              Inc(Ctr2);
              If Ctr2>30 then
                Begin
                  Inc(Ctr3);
                  If Ctr3>5 then
                    Begin
                      Inc(Ctr1);
                      Ctr3:=0;
                      If Ctr1>15 then
                        Begin
                          Ctr2:=0;
                          Ctr1:=0;
                        End;
                    End;
                End;
            End;
          4:Begin
              Inc(Ctr2);
              If Ctr2>=2 then
                Begin
                  Ctr2:=0;
                  Inc(Ctr1);
                  If Ctr1>2 then Ctr1:=1;
                End;
              Dec(Ctr3);
              If Ctr3<0 then
                Begin
                  Ctr3:=Round(Random(20))+20;
                  For B:=21 to 100 do
                    With Fire_Act[B] do
                      If Tipe=0 then
                        Begin
                          fire_Act[B].Tipe:=2;
                          X:=Object_Act[A].X+20;
                          Y:=Object_Act[A].Y+3;
                          Ctr1:=1;
                          Dx:=My_Player.X-X;
                          Dy:=My_Player.Y-Y;
                          R:=Sqrt(Dx*Dx+Dy*Dy);
```

```
                If R=0 then R:=0.000001;
                Vx:=Round(Dx*4/R);
                Vy:=Round(Dy*4/R);
                Pic:=78;
                B:=100;
              End;
        End;
    End;
5:Begin
    Inc(Ctr1);
    If Ctr1>100 then Ctr1:=1;
  End;
6:Begin
    Inc(Ctr1);
    If Ctr1>2 then Ctr1:=1;
  End;
7:Begin
    Inc(Ctr2);
    If Ctr2>3 then
      Begin
        If My_Player.X<Object_Act[A].X then
          Begin
            Direction:=0;
            Vx:=-3;
          End Else
          Begin
            Direction:=1;
            Vx:=3;
          End;
        Ctr2:=0;
        Inc(Ctr1);
        If Ctr1>2 then Ctr1:=1;
        Dec(Ctr3);
        If (Ctr3<0) then Ctr3:=0;
        If (Ctr3=0) then
          Begin
            Ctr3:=Round(Random(20))+20;
            For B:=21 to 100 do
              With Fire_Act[B] do
                If Tipe=0 then
                  Begin
                    Fire_Act[B].Tipe:=3;
                    X:=Object_Act[A].X+10;
                    Y:=Object_Act[A].Y+25;
                    Fire_Act[B].Ctr1:=1;
                    Vx:=0;
                    Vy:=4;
                    Pic:=74;
                    B:=100;
                  End;
          End;
      End;
  End;
8:Begin
```

```
                    Inc(Ctr4);
                    If Ctr4>3 then
                      Begin
                        Ctr4:=0;
                        Inc(Ctr1);
                        If Ctr1>4 then Ctr1:=1;
                        Inc(Ctr2);
                        If (Ctr2>30) and (Ctr3=0) then
                          Begin
                            Ctr1:=3;
                            Ctr3:=1;
                            Object_Act[A].Vy:=-8;
                            If My_Player.X<Object_Act[A].X then
                              Begin
                                Vx:=-4;
                                Direction:=0;
                              End Else
                              Begin
                                Vx:=4;
                                Direction:=1;
                              End;
                          End;
                      End;
                    End;
                10,11,12,13:Begin
                    Inc(Ctr4);
                    If Ctr4>3 then
                      Begin
                        Ctr4:=0;
                        Inc(Ctr1);
                        If Ctr1>3 then Ctr1:=1;
                    End;
                  End;
              End;
            End;
  End;

Procedure OutTro;
Var A:Integer;
  Begin
    Set_All_Palette(User_Pal,63,0,0);
    For A:=1 to 50 do
      Begin
        Put_Parallax_Scroll(Xp,Yp,Process^,UnComp^);
        Redraw_Screen;
        Rll_UnCompress(Picture[18+((A Mod 8) div 4)]^,UnComp^);
        T_Put3D(Process^,160-A,100-A,A*2,A*2,UnComp^,PalTable^);
        MoveTo(Process^,Screen^);
        If A Mod 10 = 0 then Metamorph_Palette(Active_Pal,;
            User_Pal,1);
        Refresh_Palette;
      End;
    Set_All_Palette(User_Pal,0,0,0);
    For A:=1 to 63 do
```

```
      Begin
        Metamorph_Palette(Active_Pal,User_Pal,1);
        Refresh_Palette;
      End;
    FillChar(Screen^,64000,0);
    Active_Pal:=Default_Pal;
    Refresh_Palette;
    Put_Center(90,8,LightBlue,'GAME OVER !!!');
    Delay(2000);
  End;

Procedure Main;
  Begin
    Repeat
      Detect_Keyboard;
      Scanning_Area;
      If My_Player.X-Xp<110 then Xp:=My_Player.X-110;
      If My_Player.Y-Yp<50 then Yp:=My_Player.Y-50;
      If My_Player.X-Xp>190 then Xp:=My_Player.X-190;
      If My_Player.Y-Yp>162 then Yp:=My_Player.Y-162;
      If Xp<0 then Xp:=0;
      If Yp<0 then Yp:=0;
      If Xp>2680 then Xp:=2680;
      If Yp>300 then Yp:=300;
      Put_Parallax_Scroll(Xp,Yp,Process^,UnComp^);
      Redraw_Screen;
      MoveTo(Process^,Screen^);
      Do_Status;
      Object_Collision;
      Do_Physics;
      If CountDown then
        Begin
          Dec(CtrCD);
          If CtrCD=0 then Finish:=True;
        End;
    Until Finish or (key=1);
    OutTro;
  End;

Procedure Removal;
  Begin
    CloseVga256;
    SetIntVec($9,OldInt9);
  End;

Begin
  Init_Device;
  Init_Environment;
  Init_Variable;
  Init_Picture;
  Init_Mission;
  Main;
  Removal;
End.
```

Cobalah untuk bermain-main sejenak dengan BUG HUNTER game ini (untuk mengistirahatkan otak sejenak sebelum kita akan membahas program kita yang terakhir ini).

Gerakkan BUG HUNTER dengan menggunakan kursor kiri-kanan dan atas untuk lompat. Tombol spacebar digunakan untuk menembak jika Anda telah memiliki tembak. Tombol ESC untuk keluar dari permainan.

Bagaimana?? VGACAD.TPU yang kita modifikasi ini canggih kan?? He-he-he. Masih banyak materi pembahasan game programming dan animasi yang jauh lebih menarik lagi tapi semuanya itu akan kita bahas pada kesempatan yang akan datang. Anda ingin tahu? Baiklah saya beri tahu sedikit deh. Materi yang mungkin akan kita bahas pada kesempatan yang akan datang ialah MODE X (resolusi 320 x 240 256 warna). Super VGA programming (1024 x 768 256 warna). Fluid Animation (animasi api dan plasma). Sprite morphing dan yang paling menarik dan cukup rumit untuk dibahas ialah 3D programming di mana kita akan mempelajari teknik-teknik pembuatan game seperti DOOM dengan teknologi Ray Casting, Wing Commander yang memanfaatkan teknologi Sprite Rotating dan Resizing, Comanche Maximum Overkill dengan teknologi Voxel Space-nya yang sampai sekarang masih dipergunakan oleh NASA dengan satelitnya untuk memvisualisasikan permukaan planet.

Dan teknologi 3D yang paling modern saat ini ialah Texture Mapping. Teknik ini dapat Anda temukan pada game Virtua Fighter 2, Descent, Terminal Velocity, High Octane, FX Fighter, Mech Warrior 2 dan lain-lainnya.

Sekarang marilah kita konsentrasi pada pembahasan engine BUG HUNTER ini. Kita tidak akan membahas seluruh procedure program ini. Tetapi kita hanya akan membahas procedure-procedure yang penting saja.

```
Uses Crt,VgaCad,Dos,VgaFont;

Const Spot     =  1;
      Shower   = 22;
      Piranha  = 20;
      Ghost    = 28;
      Cloud    = 33;
      Peer     = 36;
      Wheel    = 40;
      Snake    = 42;
      Teeth    = 44;

      P_Brown  = 47;
      P_Red    = 55;
      P_Green  = 57;
      P_Gray   = 58;
      M_Red    = 48;
      M_Green  = 49;
```

```
M_Blue  = 50;
H_Red   = 51;
H_Brown = 52;
H_Gray  = 53;
E_Box   = 56;
Mold1   = 54;
Mold2   = 59;

Bomb    = 88;
Misile  = 91;
Coin    = 92;
Invul   = 95;
```

Konstanta yang sama digunakan seperti pada EDITOR dari BUG HUNTER ini.

```
Type Enemy_Rec = Record
       X,Y    : Integer;
       Tipe   : Byte;
       Energy : Integer;
     End;
```

Enemy_Rec ini adalah record yang digunakan untuk menyimpan data objek.

```
Block_Rec = Record
  X,Y    : Integer;
  Pic    : Byte;
End;
```

Block_Rec ini adalah record yang digunakan untuk menyimpan data balok.

```
Char_Rec = Record
  Length,Width: Word;
  Picture     : Integer;
End;
```

Record di atas ini digunakan untuk menyimpan data gambar objek.

```
Obj_Rec = Record
  Obj_Tagged: Integer;
  X,Y       : Integer;
  Vx,Vy     : Integer;
  Ctr1,Ctr2,
  Ctr3,Ctr4,Ctr5
            : Integer;
  Direction : Byte;
End;
```

Record OBJ_REC ini ialah struktur data objek yang diakses oleh program. Adapun objek-objek ini ialah objek yang lokasinya berdekatan dengan si Buggy.

- Obj_Tagged — ialah nomor objek yang diakses oleh record ini.
- X, Y — ialah koordinat objek.
- Vx, Vy — ialah kecepatan sumbu X dan sumbu Y objek.
- Ctr1-Ctr 5 — ialah variabel Counter yang memiliki fungsi yang berbeda untuk tiap jenis Objek.
- Direction — ialah arah objek, 1 = arah kanan, 0 = arah kiri.

```
Fire_Rec = Record
  X, Y       : Integer;
  Vx,Vy      : Integer;
  Pic        : Integer;
  Ctr1,Ctr2,Ctr3,
  Ctr4,Ctr5  : Integer;
  Direction  : Byte;
  Destruct   : Byte;
  Tipe       : Byte;
End;
```

FIRE_REC ini adalah struktur data peluru yang akan ditembakkan baik oleh Buggy ataupun oleh musuh.

- X, Y — ialah koordinat peluru.
- Vx, Vy — ialah kecepatan peluru.
- Pic — ialah nomor gambar peluru.
- Ctr1-Ctr5 — ialah variabel Counter.
- Direction — menentukan arah peluru.
- Destruct — tidak terpakai.
- Tipe — menentukan jenis peluru.

```
Player_Rec = Record
  Active     : Boolean;
  X,Y        : Integer;
  Vx,Vy      : Integer;
  Direction  : Byte;
  Status     : Byte;
  Life       : Byte;
  FCtr       : Integer;
  Ctr1,Ctr2,Ctr3,
  Ctr4,Ctr5  : Integer;
  Score      : Integer;
  PowerUp    : Boolean;
  Misile     : Integer;
  InVul      : Boolean;
```

```
    InCtr      : Integer;
  End;
```

Record PLAYER_REC ini ialah struktur data si Buggy.

- Active — Berharga TRUE jika Buggy masih hidup.
- X, Y — Koordinat si Buggy.
- Vx, Vy — Kecepatan si Buggy.
- Direction — Arah si Buggy.
- Status — Status si Buggy (status menentukan apakah si Buggy sedang berjalan, melompat, jatuh, dan sebagainya).
- Life — Jumlah nyawa yang tersisa.
- Fctr — Frame Counter.
- Ctr1-Ctr5 — Counter.
- Score — Nilai yang telah diperoleh si Buggy.
- PowerUp — TRUE jika si Buggy telah memiliki misile launcher.
- Misile — Jumlah rocket yang dimiliki si Buggy.
- InVul — TRUE jika si Buggy mendapat bonus kekebalan.
- InCtr — Counter untuk menentukan habisnya kekebalan yang diperoleh oleh si Buggy.

```
P_Rec = Record
  Length,Width:Word;
End;
```

Record yang berisi panjang dan lebar tiap gambar.

```
Mess_Rec = Record
  St:String;
  X,Y:Integer;
  Active:Boolean;
  Col:Byte;
  Ctr:Byte;
End;
```

MESS_REC adalah record yang akan digunakan untuk menyimpan data text yang akan ditampilkan pada layar monitor.

- St — Variabel string yang menyimpan tulisan yang akan ditampilkan,
- X, Y — Koordinat X dan Y dari tulisan.
- Active — TRUE jika tulisan masih harus ditampilkan pada layar monitor.
- Col — Warna tulisan.
- Ctr — Variabel counter yang menentukan apakah tulisan tersebut sudah cukup lama keluar di layar monitor.

```
Anim_Rec = Record
  X,Y  : Integer;
  Ctr  : Integer;
  Tipe : Byte;
  Active: Boolean;
End;
```

ANIM_REC ialah record untuk menyimpan animasi yang akan dimunculkan pada layar monitor.

- X, Y — ialah koordinat X dan Y dari animasi.
- Ctr — ialah counter frame animasi.
- Tipe — ialah jenis animasi.
- Active — TRUE jika animasi tersebut masih harus ditampilkan pada layar monitor.

```
Var Fil         : File;
    Atr         : Text;
    A           : Integer;
    Picture     : Array[1..200] of Pointer;
    PicSize     : Array[1..200] of P_Rec;
    Finish      : Boolean;
    Ch          : Char;
    Szs         : Word;
    Object_Char: Array[1..20] of Char_Rec;
    Enemy_Spot  : Array[1..200] of  Enemy_Rec;
    Anim_Act    : Array[1..100] of Anim_Rec;
```

Anim_Act ialah variabel array yang akan kita gunakan untuk menampung semua animasi yang ada di layar. Animasi ini ialah objek pasif seperti yang telah dijelaskan pada awal bab ini.

```
    Block      : Array[1..1000] of Block_Rec;
    Object_Act : Array[1..40] of Obj_Rec;
```

Object_Act ialah array yang akan kita gunakan untuk menampung semua objek yang berdekatan dengan si Buggy.

```
Block_Act   : Array[1..200] of Integer;
```

Block_Act ialah array yang akan kita gunakan untuk menampung semua balok yang berdekatan dengan si Buggy.

```
Fire_Act    : Array[1..100] of Fire_Rec;
```

Fire_Act ialah array yang akan kita gunakan untuk menampung peluru yang ada di layar.

Adapun Fire_Act nomor 1-20 untuk peluru si Buggy dan nomor 21-100 untuk peluru musuh.

```
Message     : Array[1..20] of Mess_Rec;
```

Message menyimpan tulisan-tulisan yang akan ditampilkan pada layar monitor.

```
CountDown   : Boolean;
CtrCD       : Integer;
```

Kedua variabel di atas digunakan pada saat game telah selesai sehingga program tidak langsung keluar dari permainan tetapi menunggu sebanyak CtrCD loop.

```
My_Player   : Player_Rec;
```

Variabel yang menampung data-data si Buggy.

```
Mission_File: File;
Process     : Pointer;
UnComp      : Pointer;
Xp,Yp       : Integer;
OldInt9     : Pointer;
PalTable    : Pointer;
Key         : Byte;
Flag,Flag1,Flag2,Flag3,Flag4,Flag5,Flag6:Boolean;
```

Variabel-variabel di atas ialah variabel yang untuk proses virtual screen, picture un-kompres, Xp dan Yp yang digunakan untuk koordinat sprite parallax scrolling. Juga untuk tabel transparant, dan interrupt keyboard yang diganti. Semuanya itu sama persis dengan program-program yang telah kita bahas.

```
Function Collision(No:Byte;X,Y:Integer;Xl,Yl:Integer):Boolean;
Var A:Integer;
  Begin
    A:=1;
```

Function ini digunakan untuk mendeteksi apakah terjadi tabrakan antara sebuah objek dengan balok.

```
Collision:=False;
If (Y+YL>462) then Collision:=True else
```

Jika objek tersebut menembus permukaan tanah (koordinat Y = 462) maka tabrakan terjadi.

```
Repeat
  If (X+XL>Block[Block_Act[A]].X+2) and;
     (X<Block[Block_Act[A]].X+12) and
     (Y+YL>Block[Block_Act[A]].Y) and;
     (Y<Block[Block_Act[A]].Y+9) then
       Begin
         Collision:=True;
         A:=200;
       End;
  Inc(A);
Until (Block_Act[A]=0) or (A=201);
```

Jika tidak, cari balok yang menabrak objek tersebut. Untuk ini kita cukup memeriksa balok yang berada pada array BLOCK_ACT karena kita hanya cukup memeriksa balok yang dekat dengan si BUGGY.

```
  End;

Procedure Make_Anim(No:Byte;Xs,Ys:Integer);
Var A:Integer;
  Begin
    For A:=1 to 100 do
      With Anim_Act[A] do
        If not Active then
          Begin
            Tipe:=No;
            Active:=True;
            Ctr:=1;
            X:=Xs;Y:=Ys;
            A:=100;
          End;
```

Aktifkan sebuah animasi dengan tipe yang diberikan lewat parameter. Procedure ini dipanggil pada saat terjadinya percikan api, munculnya asap dan lain-lain yang membutuhkan gambar animasi pasif.

```
  End;

Procedure Create_Message(Sts:String;Xs,Ys:Integer;Cols:Byte);
Var A:Integer;
  Begin
    For A:=1 to 20 do
      If Message[A].Active=False then
```

```
        With Message[A] do
          Begin
            X:=Xs;Y:=Ys;
            St:=Sts;Col:=Cols;
            Active:=True;
            Ctr:=1;
            A:=20;
          End;
```

Procedure ini bertujuan untuk mengaktifkan sebuah message yaitu jika si Buggy kehilangan nyawa, mendapat score, mendapat misile launcher dan lain-lainnya. Setelah message ini diaktifkan maka secara otomatis bagian message ini akan ditampilkan ke layar oleh procedure REDRAW_SCREEN.

```
  End;

Procedure Init_Environment;
  Begin
    OpenVga256;
    LoadFont(@Small);
    Install_Palette('Palette.Pal');
    Load_Parallax_Scroll('Back1');
    GetMem(PalTable,65280);
    Install_Transparent(PalTable,'Trans.Tab');
```

Instal segala keperluan program ini, seperti tabel transparant, font, mode graphic, parallax scrolling dan palette.

```
  End;

Procedure Init_Variable;
  Begin
    CtrCD:=30;
    CountDown:=False;
```

Set CtrCD = 30 sehingga jika terjadi game over, program masih akan melakukan loop sebanyak 30 kali.

```
    Xp:=0;Yp:=300;
    GetMem(Process,64000);
    GetMem(UnComp,10000);
    FillChar(Block,5000,0);
    FillChar(Enemy_Spot,1400,0);
    FillChar(Object_Act,40*SizeOf(Obj_Rec),0);
    FillChar(Fire_Act,100*SizeOf(Fire_Rec),0);
    FillChar(Block_Act,200*SizeOf(Block_Rec),0);
    FillChar(Anim_Act,100*SizeOf(Anim_Rec),0);
    FillChar(Message,20*SizeOf(Mess_Rec),0);
```

Inisialisasikan semua variabel array yang akan kita gunakan.

```
    With My_Player do
      Begin
        InVul:=False;
        InCtr:=0;
        Misile:=0;
        Life:=6;
        PowerUp:=False;
        Score:=0;
        Active:=True;
        X:=50;
        Y:=400;
```

Letakkan player kita pada koordinat (50,400) . Sebenarnya hal ini tidak perlu karena toh nantinya koordinat player kita akan dibaca dari file MISSION.001. Inisialisasikan variabel yang lain.

```
        Vx:=0;Vy:=0;
        Ctr1:=0;Ctr2:=0;Ctr3:=1;Ctr4:=0;Ctr5:=1;
        Status:=1;
        Direction:=1;
        FCtr:=1;
      End;
  End;

Procedure Init_Device;
  Begin
    Randomize;
    GetIntVec($9,OldInt9);
    SetIntVec($9,@Int9);
    Flag:=False;
    Flag1:=False;
    Flag2:=False;
    Flag3:=False;
    Flag4:=False;
    Flag5:=False;
    Flag6:=False;
```

Siapkan interrupt keyboard yang baru.

```
  End;

Procedure Init_Mission;
  Begin
    Assign(Mission_File,'Pics\Mission.001');
    Reset(Mission_File,1);
    BlockRead(Mission_File,Object_Char,20*SizeOf(Char_Rec));
    BlockRead(Mission_File,Enemy_Spot,200*SizeOf(Enemy_Rec));
    BlockRead(Mission_File,Block,1000*SizeOf(Block_Rec));
    Close(Mission_File);
```

Baca data level.

```
    For A:=1 to 1000 do
      If Enemy_Spot[A].Tipe=1 then
        Begin
          My_Player.X:=Enemy_Spot[A].X;
          My_Player.Y:=Enemy_Spot[A].Y;
          A:=1000;
        End;
```

Cari objek yang bertipe si Buggy dan ambil koordinat X dan Y-nya untuk dimasukkan ke dalam variabel koordinat player kita. Dengan cara ini kita dapat meletakkan posisi start player kita di manapun juga.

```
  End;

Procedure Detect_Keyboard;
  Begin
  With My_Player do
   If (Status<>4) and (Life>0) then
   Begin
```

Jika masih hidup dan status <>4. Status nomor 4 ialah status di mana si Buggy sedang terlempar karena menabrak musuh. Pada saat itu si Buggy tidak dapat dikendalikan sesaat.

```
    If Flag then
      Begin
        Vx:=Vx-1;
        If Status<>3 then Status:=2;
        If Vx<-6 then Vx:=-6;
        Direction:=0;
```

Jika tombol kiri ditekan, kurangi Vx yang berfungsi sebagai variabel kecepatan horizontal.

Jika Status <>3 yang berarti si Buggy tidak dalam kondisi melompat maka set Status ke 2 yang menandakan bahwa si Buggy sedang berjalan. Hal ini dilakukan karena pada saat melompat, si Buggy masih bisa dikendalikan. Karena kita tidak ingin menampilkan gambar si Buggy sedang berjalan bila kita menekan tombol samping pada saat si Buggy melompat, maka tidak akan mengubah Status menjadi 2.

Batasi terminal velocity pada -6.
Direction = 0 menunjukkan si Buggy sedang menghadap ke kiri.

```
      End;
    If (Flag2) then
      Begin
        Vx:=Vx+1;
        If Status<>3 then Status:=2;
        If Vx>6 then Vx:=6;
        Direction:=1;
```

Lakukan hal yang sama seperti di atas. Cuma di sini arah gerakan si Buggy ke kanan.

```
    End;
  If Flag3 then
    Begin
      If (Status=3) and (Ctr1<12) then
        Begin
          Vy:=Vy-1;
          Status:=3;
        End;
```

Jika tombol lompat ditekan dan status si Buggy sedang melompat dan Ctr1 < 12 maka kurangi kecepatan vertikal yang berarti si Buggy melompat ke atas dengan kecepatan konstan (karena setelah ini Vy akan ditambah dengan 1 oleh pengaruh teknik gravitasi). Set Status = 3 yang menunjukkan si Buggy sedang dalam status lompat. Digunakan testing Ctr1 < 12 untuk membuat si Buggy dapat melompat dengan berbagai ketinggian tergantung dengan lamanya tombol atas Anda tahan. Karena setelah Ctr1 >=12 kecepatan sumbu Y tidak akan dikurangi lagi sehingga si kecepatan Vy akan bertambah karena pengaruh gravitasi dan mengakibatkan kecepatan lompat si buggy makin berkurang hingga akhirnya si Buggy jatuh kembali ke tanah. Jadi Anda masih dapat menambah kecepatan lompatan si Buggy selama 12 loop (kurang lebih sekitar 0.5 detik).

```
      If (Status<>3) then
        Begin
          Vy:=-5;
          Status:=3;
          Ctr1:=0;
        End;
```

Jika si buggy tidak dalam keadaan melompat maka set Status = 3 dan Set Ctr1 = 0 untuk memberikan kesempatan bagi pemain dalam menahan kecepatan awal selama 12 loop. Kemudian berikan kecepatan awal sebesar -5 untuk melompat ke atas.

```
    End;

  If Flag6 and (My_Player.Ctr3=0) and My_Player.PowerUp and;
     (My_Player.Misile>0) then
```

Jika tombol tembak ditekan dan Ctr3 =0 yang menunjukkan waktu di mana si Buggy sudah dapat melakukan tembakan lagi dan si Buggy telah mendapat misile launcher dan masih memiliki roket yang cukup maka lakukan proses menembak.

```
      Begin
        Dec(My_Player.Misile);
```

Kurangi jumlah roket.

```
For A:=1 to 20 do
  With Fire_Act[A] do
    If Tipe=0 then
      Begin
        Tipe:=1;
        X:=My_Player.X+10;
        Y:=My_Player.Y+4;
        If My_Player.Direction=1 then Vx:=7 else Vx:=-7;
```

Set kecepatan roket sesuai dengan arah si Buggy.

```
        My_Player.Ctr3:=20;
```

Set Ctr3 menjadi 20 yang berarti si Buggy tidak akan dapat menembak selama selang waktu 20 loop.

```
        My_Player.FCtr:=1;
        My_Player.Status:=5;
```

Set Status = 5 untuk menampilkan gambar si Buggy sedang menembak. Dan set Frame Counter = 1 karena terjadi action baru bagi si Buggy.

```
        Pic:=67;
        Ctr1:=0;
        Vy:=0;
        A:=20;
```

Isikan nomor gambar dari peluru ke gambar peluru si Buggy yaitu nomor 67.

```
      End;
    End;
End;
  End;

Procedure Enemy_Startup(A,B:Integer);
  Begin
```

Procedure ini digunakan untuk memberikan harga awal dari tiap musuh yang baru pertama kali muncul pada layar monitor. Adapun jenis musuh yang berbeda memiliki setting yang berbeda pula.

```
  With Object_Act[B] do
    Case Enemy_Spot[A].Tipe of
      2:;
```

Musuh nomor 2 tidak terpakai.

```
3:Begin
    Vx:=0;
    Vy:=0;
    Ctr2:=0;
    Ctr1:=0;
    Ctr3:=0;
    Ctr5:=0;
    Direction:=1;
  End;
```

Musuh nomor 3 ialah Piranha Plant. Karena dia tidak bergerak maka set Vx dan Vy = 0.

```
4:Begin
    Vx:=1;
    Vy:=0;
    Ctr1:=1;
    Ctr2:=1;
    Ctr5:=0;
    Direction:=1;
```

Musuh nomor 4 ialah Shower Man. Pada saat dia muncul dia akan langsung bergerak ke kanan dengan kecepatan 1 pixel per loop. Karena itu Vx di-set = 1.

```
  End;
5:Begin
    If My_Player.X<Enemy_Spot[A].X then Vx:=-3 else Vx:=3;
    Vy:=-8;
    If Vx<0 then Direction:=0 else Direction:=1;
    Ctr1:=1;Ctr2:=1;
    Ctr5:=0;
```

Musuh nomor 5 ialah Peer Doll. Gerakan dari Peer Doll ini ialah ke arah si Buggy dan selalu melompat-lompat sehingga kita harus mengisi kecepatan sumbu X dengan arah yang menuju si Buggy. Sedangkan kecepatan vertikal diset -8 yang menunjukkan kecepatan awal si Peer Doll ialah 8 pixel ke atas.

```
  End;
6:Begin
    Vx:=-2;
    Vy:=0;
    Ctr1:=1;
    Direction:=1;
    Ctr5:=0;
```

Musuh nomor 6 ialah Red Wheel. Gerakan dari Red Wheel ini ialah ke arah kiri dengan kecepatan 2 pixel per loop.

```
  End;
7:Begin
    If My_Player.X<Enemy_Spot[A].X then Vx:=-3 else Vx:=3;
```

```
            Vy:=0;
            If Vx<0 then Direction:=0 else Direction:=1;
            Ctr1:=1;
            Ctr2:=0;
            Ctr3:=0;
            Ctr5:=0;
```

Musuh nomor 7 ialah Cloud yang akan bergerak ke arah kita, untuk itu Vx diarahkan menuju si buggy dengan magnitude sebesar 3 pixel per loop.

```
          End;
        8:Begin
            Vx:=0;
            Vy:=0;
            Ctr1:=1;
            Ctr2:=0;
            Ctr3:=0;
            Ctr4:=0;
            Ctr5:=0;
            Direction:=1;
```

Musuh nomor 8 ialah Teeth yang pada start awalnya diam tidak berpindah tempat. Untuk itu kecepatan awalnya di set = 0.

```
          End;
        10..13:Begin
            Vx:=0;
            Vy:=0;
            Ctr1:=1;
            Ctr2:=0;
            Ctr3:=0;
            Ctr4:=0;
            Ctr5:=0;
            Direction:=1;
```

Jenis objek ini ialah jenis bonus. Maka tidak perlu diisikan kecepatan awalnya.

```
          End;
      End;
  End;

Function GetPicture(No,Ctr,Direction:Integer):Integer;
  Begin
    GetPicture:=0;
```

Procedure ini bertujuan untuk memberikan nomor gambar dari sebuah objek dengan arah dan nomor frame yang diinputkan lewat parameter.

```
    Case No of
      2:Begin;
```

```
  End;
3:Case Ctr of
    1:GetPicture:=27;
    2:GetPicture:=26;
    3:GetPicture:=25;
    4:GetPicture:=24;
    5:GetPicture:=23;
    6:GetPicture:=22;
    7:GetPicture:=23;
    8:GetPicture:=22;
    9:GetPicture:=23;
    10:GetPicture:=22;
    11:GetPicture:=23;
    12:GetPicture:=24;
    13:GetPicture:=25;
    14:GetPicture:=26;
    15:GetPicture:=27;
  End;
4:Begin
  If Direction=1 then
    Case Ctr of
      1:GetPicture:=20;
      2:GetPicture:=21;
    End Else
    Case Ctr of
      1:GetPicture:=119;
      2:GetPicture:=120;
    End;
  End;
5:Begin
  Case Ctr of
    1:GetPicture:=39;
    2:GetPicture:=38;
    3:GetPicture:=37;
    4..100:GetPicture:=36;
  End;
  End;
6:Begin
    Case Ctr of
      1:GetPicture:=40;
      2:GetPicture:=41;
    End;
  End;
7:Begin
  If Direction=1 then
    Case Ctr of
      1:GetPicture:=33;
      2:GetPicture:=34;
    End Else
    Case Ctr of
      1:GetPicture:=121;
      2:GetPicture:=122;
    End;
  End;
```

```
      8:Begin
        If Direction=0 then
          Case Ctr of
            1:GetPicture:=44;
            2:GetPicture:=45;
            3:GetPicture:=46;
            4:GetPicture:=45;
          End Else
          Case Ctr of
            1:GetPicture:=126;
            2:GetPicture:=127;
            3:GetPicture:=128;
            4:GetPicture:=127;
          End;
        End;
      10:Begin
          Case Ctr of
            1:GetPicture:=86;
            2:GetPicture:=87;
            3:GetPicture:=88;
            4:GetPicture:=87;
          End;
        End;
     11:Begin
          Case Ctr of
            1:GetPicture:=89;
            2:GetPicture:=90;
            3:GetPicture:=91;
            4:GetPicture:=90;
          End;
        End;
      13:Begin
          Case Ctr of
            1:GetPicture:=92;
            2:GetPicture:=93;
            3:GetPicture:=94;
            4:GetPicture:=93;
          End;
         End;
      12:Begin
          Case Ctr of
            1:GetPicture:=95;
            2:GetPicture:=96;
            3:GetPicture:=97;
            4:GetPicture:=96;
          End;
        End;
    End;
  End;
Procedure Scanning_Area;
Var A,B:Integer;
    Cancel:Boolean;
    St:String;
  Begin
```

Procedure ini bertujuan untuk memasukkan objek dan balok yang dekat dengan si Buggy ke dalam variabel buffer. Sehingga untuk proses selanjutnya kita tidak perlu memproses seluruh objek dan balok yang ada.

```
For A:=1 to 40 do
 IF Object_Act[A].Obj_Tagged<>0 then
  With Enemy_Spot[Object_Act[A].Obj_Tagged] do
    Begin
      If (Object_Act[A].X+Object_Char[Tipe].Length<Xp) or
      (Object_Act[A].X>Xp+319) or;
        (Object_Act[A].Y+Object_Char[Tipe].Width<Yp)
      or (Object_Act[A].Y>Yp+199) then
```

Jika objek tersebut telah berada di luar layar maka hilangkan objek tersebut dari buffer, pada jenis objek tertentu hilangkan juga objek tersebut dari data objek. Dengan cara ini ada sebagian dari musuh yang setelah melewati batas layar tidak akan muncul kembali. Tetapi ada juga objek yang setelah melewati batas layar kemudian muncul kembali pada start awal (misalnya si Cloud dan si Piranha Plant). Perhatikan listing di bawah ini. Case Tipe of digunakan untuk menyeleksi jenis objek. Jika dilakukan Tipe:= Tipe maka objek tersebut akan muncul lagi setelah hilang dari batas layar. Tetapi jika dilakukan Tipe: = 0 maka objek tersebut akan hilang selamanya dari permainan. Perhatikan untuk objek nomor 3 dan nomor 7 di mana dilakukan proses Tipe:= Tipe. Kedua objek tersebut ialah Piranha Plant dan Cloud. Mereka akan muncul lagi setelah hilang dari layar.

```
            If Tipe >1 then
              Begin
                Object_Act[A].Obj_Tagged:=0;
                Case Tipe of
                  2:;
                  3:Tipe:=Tipe;
                  4:Tipe:=0;
                  5:Tipe:=0;
                  6:Tipe:=0;
                  7:Tipe:=Tipe;
                  8:Tipe:=0;
                  9..12:Tipe:=Tipe;
                End;
              End;
    End;
For A:=1 to 200 do
  With Enemy_Spot[A] do
    Begin
      If (X+Object_Char[Tipe].Length>Xp) and (X<Xp+319) and
         (Y+Object_Char[Tipe].Width>Yp)  and (Y<Yp+199) and ;
          (Tipe >1) then
          Begin
```

Jika objek tersebut berada dekat dengan si Buggy maka masukkan objek tersebut ke dalam buffer objek yang akan diproses yaitu OBJEK_ACT.

```
Cancel:=False;
For B:=1 to 40 do
  Begin
    If Object_Act[B].Obj_Tagged=A then
      Begin
        Cancel:=True;
        B:=40;
      End;
  End;
```

Tetapi sebelumnya periksa terlebih dahulu apakah objek tersebut telah berada di buffer.

```
If Not Cancel then
```

Jika belum ada maka ...

```
For B:=1 to 40 do
  If Object_Act[B].Obj_Tagged=0 then
    Begin
        Object_Act[B].Obj_Tagged:=A;
        Object_Act[B].X:=X;
        Object_Act[B].Y:=Y;
        Enemy_Startup(A,B);
        B:=40;
```

Masukkan objek tersebut ke dalam buffer dan panggil ENEMY_STARTUP untuk melakukan inisialisasi dari objek tersebut.

```
                End;
            End;
        End;
    B:=0;
    For A:=1 to 1000 do
      With Block[A] do
        If Pic<>0 then
          Begin
            If (X<Xp+379) and (X+16>Xp-50) and (Y<Yp+260) and;
                (Y+11>Yp-60) then
              Begin
                Inc(B);
                Block_Act[B]:=A;
              End;
          End;
    Inc(B);
    Block_Act[B+1]:=0;
End;
```

Lakukan proses yang sama terhadap balok. Masukkan balok yang dekat dengan si Buggy ke dalam buffer yaitu BLOCK_ACT.

```
Procedure Redraw_Screen;
Var A,Pic:Integer;
  Begin
    For A:=1 to 100 do
      With Fire_Act[A] do
        If Tipe<>0 then
          Begin
            Put(Process^,X-Xp,Y-Yp,Pic);
          End;
```

Gambar seluruh peluru yang ada.

```
    For A:=1 to 20 do
      With Message[A] do
        If Active then
          Begin
            If (Y>Yp+1) and (Y+8<Yp+199) then PutString(Process^,;
                X-Xp,Y-Yp,1,Col,St);
          End;
```

Gambar seluruh message yang ada selama message tersebut masih dalam penglihatan layar.

```
    For A:=1 to 200 do
      If Block_Act[A]=0 then
        Begin
          A:=200;
        End Else
        With Block[Block_Act[A]] do
          Begin
            Put(Process^,X-Xp,Y-Yp,Pic);
          End;
```

Gambar seluruh balok yang ada pada virtual screen.

```
    For A:=1 to 100 do
      With Anim_Act[A] do
        If Active then
          Begin
            Case Tipe of
              1:Begin
                  Case Ctr of
                    1..3:Pic:=70;
                    4..6:Pic:=71;
                    7..9:Pic:=72;
                    10..12:Pic:=73;
```

Animasi tipe ini ialah percikan api ledakan roket si Buggy.

```
      End;
    End;
2:Begin
    Case Ctr of
      1..3:Pic:=79;
      4..6:Pic:=80;
```

Animasi tipe ini ialah animasi busa sabun yang peah.

```
      End;
    End;
3:Begin
    Case Ctr of
      1..3:Pic:=74;
      4..6:Pic:=75;
      7..9:Pic:=76;
      10..12:Pic:=77;
```

Animasi ini ialah animasi air yang menetes ke lantai.

```
      End;
    End;
4:Begin
    Case Ctr of
      1..3:Pic:=61;
      4..6:Pic:=62;
      7..9:Pic:=63;
      10..12:Pic:=61;
```

Animasi ini ialah animasi ledakan merah di mana terjadinya kematian sebuah musuh.

```
      End;
    End;
5:Begin
    Case Ctr of
      1..3:Pic:=85;
      4..6:Pic:=84;
      7..9:Pic:=83;
      10..12:Pic:=82;
      13..15:Pic:=81;
```

Animasi tipe ini ialah animasi asap.

```
      End;
    End;
6:Begin
    Case Ctr of
      1..2:Pic:=64;
      3..4:Pic:=65;
      5..6:Pic:=66;
      7..8:Pic:=65;
    End;
```

```
            X:=My_Player.X-10;Y:=My_Player.Y-10;
          End;
```

Animasi tipe 6 ini ialah animasi bola kekebalan. Karena bola ini harus selalu berada di tengah-tengah si Buggy maka ambil koordinat X dan Y dari Buggy sebagai koordinat dari animasi ini.

```
        End;
        If Tipe in[5,6] then PutTrans(Process^,X-Xp,;
            Y-Yp,Pic) else
          Put(Process^,X-Xp,Y-Yp,Pic);
```

Jika tipe animasi tersebut ialah asap atau kekebalan maka letakkan sprite dengan mode transparant.

```
      End;
```

Gambar seluruh animasi yang ada pada virtual screen.

```
  For A:=1 to 40 do
    If Object_Act[A].Obj_Tagged<>0 then
      With Object_Act[A] do
        Begin

Pic:=GetPicture(Enemy_Spot[Obj_Tagged].Tipe,Ctrl,Direction);
        If Pic<>0 then
          If Ctr5=0 then Put(Process^,X-Xp,Y-Yp,Pic) else
          PutTrans(Process^,X-Xp,Y-Yp,Pic);
```

Gambar seluruh musuh pada virtual screen. Jika CTR5<>0 yang berarti musuh tersebut sedang dalam proses mati, maka letakkan sprite dengan mode transparant.

```
        End;
  With My_Player do
   Begin
   If PowerUp then
    Begin
```

Jika si Buggy sedang memiliki misile launcher, gunakan gambar-gambar di bawah ini.

```
    If Direction=1 then
      Begin
        Case Status of
          1:Pic:=12;
```

Si buggy sedang diam. (Hadap kanan.)

```
          2:Case FCtr of
              1..3:Pic:=11;
```

```
        4..6:Pic:=12;
        7..9:Pic:=13;
        10..12:Pic:=14;
        13..15:Pic:=13;
        16..18:Pic:=12;
```

Si Buggy sedang berjalan. (Hadap kanan.)

```
      End;
    3:Case FCtr of
        1:Pic:=5;
        2:Pic:=6;
        3:Pic:=7;
        4:Pic:=8;
        5:Pic:=9;
        6..200:Begin
                 If FCtr Mod 2 = 1 then Pic:=9 else Pic:=10;
               End;
```

Si Buggy sedang lompat. (Hadap kanan.)

```
      End;
    4:Case FCtr of
        1..100:Pic:=18 + FCtr Mod 2;
```

Si Buggy sedang terlempar karena tertabrak/tertembak musuh. (Hadap kanan.)

```
      End;
    5:Case FCtr of
        1:Pic:=15;
        2:Pic:=15;
        3:Pic:=16;
        4:Pic:=16;
      End;
```

Si buggy sedang menembak. (Hadap kanan.)

```
    End;
  End Else
  Begin
```

Jika si Buggy sedang menghadap kiri maka gunakan gambar-gambar di bawah ini.

```
  Case Status of
    1:Pic:=109;
    2:Case FCtr of
        1..3:Pic:=108;
        4..6:Pic:=109;
        7..9:Pic:=110;
        10..12:Pic:=111;
        13..15:Pic:=110;
```

```
            16..18:Pic:=109;
          End;
        3:Case FCtr of
            1:Pic:=102;
            2:Pic:=103;
            3:Pic:=104;
            4:Pic:=105;
            5:Pic:=106;
            6..200:Begin
                     If FCtr Mod 2 = 1 then Pic:=107 else;
                        Pic:=106;
                   End;
          End;
        4:Case FCtr of
            1..100:Pic:=18 + FCtr Mod 2;
          End;
        5:Case FCtr of
            1:Pic:=112;
            2:Pic:=112;
            3:Pic:=113;
            4:Pic:=113;
          End;
      End;
    End;
End Else
Begin
```

Jika si Buggy tidak memiliki misile launcher, gunakan gambar-gambar di bawah ini.

```
If Direction=1 then
  Begin
    Case Status of
      1:Pic:=1;
      2:Case FCtr of
          1..3:Pic:=2;
          4..6:Pic:=3;
          7..9:Pic:=4;
          10..12:Pic:=3;
          13..15:Pic:=2;
          16..18:Pic:=1;
        End;
      3:Case FCtr of
          1:Pic:=5;
          2:Pic:=6;
          3:Pic:=7;
          4:Pic:=8;
          5:Pic:=9;
          6..200:Begin
                   If FCtr Mod 2 = 1 then Pic:=9 else Pic:=10;
                 End;
        End;
      4:Case FCtr of
          1..100:Pic:=18 + FCtr Mod 2;
```

```
          End;
        End;
      End Else
      Begin
        Case Status of
          1:Pic:=98;
          2:Case FCtr of
              1..3:Pic:=99;
              4..6:Pic:=100;
              7..9:Pic:=101;
              10..12:Pic:=100;
              13..15:Pic:=99;
              16..18:Pic:=98;
            End;
          3:Case FCtr of
              1:Pic:=102;
              2:Pic:=103;
              3:Pic:=104;
              4:Pic:=105;
              5:Pic:=106;
              6..200:Begin
                       If FCtr Mod 2 = 1 then Pic:=107 else;
                          Pic:=106;
                     End;
            End;
          4:Case FCtr of
              1..100:Pic:=18 + FCtr Mod 2;
            End;
        End;
       End;
      End;
     If Active then
       If Ctr5>30 then Put(Process^,X-Xp,Y-Yp,Pic) else
       PutTrans(Process^,X-Xp,Y-Yp,Pic);
    End;
```

Jika si Buggy sedang dalam masa 3 detik kebal setelah kehilangan nyawa, letakkan sprite dengan mode transparant.

```
PutString(Process^,230,2,0,15,'SCORE: '+Strig(My_Player.Score));
PutString(Process^,130,2,0,LightGreen,;
    'MISSILE: '+Strig(My_Player.Misile));
```

Tulis score dan jumlah rocket yang dipunyai si buggy ke layar.

```
Rll_UnCompress(Picture[1]^,UnComp^);
For A:=1 to My_Player.Life do
  Begin
    T_Put3D(Process^,5+(A-1)*20,2,16,16,UnComp^,PalTable^);
  End;
```

Letakkan gambar si Buggy pada sisi kiri atas layar yang menunjukkan jumlah nyawa yang tersisa.

Untuk ini digunakan teknik 3D transparant sehingga gambar yang dihasilkan agak kecil dan tidak mengganggu penglihatan pemain.

```
  End;

Function Check_Move(No:Byte;Xr,Yr:Integer;
        Var Vx,Vy,Rx,Ry:Integer;Xl,Yl:Integer):Byte;
Var A:Integer;
    Vrx,Vry:Real;
    Finish:Boolean;
    X,Y:Real;
    Loop,Count:Integer;
  Begin
    X:=Xr;Y:=Yr;
```

Anda tentunya familiar dengan procedure ini kan? Ini adalah procedure yang sama yang kita pergunakan pada pengecekan tabrakan dengan sistem vektor.

Untuk itu procedure ini tidak akan dijelaskan lebih lanjut.

```
    If Abs(Vx)>Abs(Vy) then
      Begin
        If Vx>0 then Vrx:=1 else Vrx:=-1;
        Loop:=Abs(Vx);
        Vry:=Vy/Abs(Vx);
      End Else
    If Vy<>0 then
      Begin
        If Vy>0 then Vry:=1 else Vry:=-1;
        Vrx:=Vx/Abs(Vy);
        Loop:=Abs(Vy);
      End Else
      Begin
        Vry:=0;Vrx:=0;
        Loop:=1;
      End;
    Finish:=False;
    Count:=0;
    Check_Move:=0;
    Repeat
      Inc(Count);
      If Collision(1,Round(X+Vrx),Round(Vry+Y),Xl,Yl)=False then
        Begin
          X:=X+Vrx;
          Y:=Y+Vry;
        End Else
        Begin
          If Collision(1,Round(Vrx+X),Round(Y),Xl,Yl)=False then
```

```
            Begin
              { Height Collision }
              Finish:=True;
              If Vy>=0 then Check_Move:=2 else Check_Move:=4;
            End Else
          If Collision(1,Round(X),Round(Vry+Y),Xl,Yl)=False then
            Begin
              { Side Collision }
              Finish:=True;
              Check_Move:=1
            End Else
            Begin
              { Both Collision }
              Finish:=True;
              Check_Move:=3;
            End;
        End;
      If Count>=Loop then Finish:=True;
    Until Finish;
    Rx:=Round(X)-Xr;
    Ry:=Round(Y)-Yr;
  End;

Procedure Do_Physics;
Var Rx,Ry: Integer;
    A    : Integer;
    XL,YL: Integer;
    Tipe : Byte;
  Begin
```

Pada procedure ini semua perhitungan perpindahan objek dilakukan.

```
    For A:=1 to 100 do
      With Fire_Act[A] do
        If (Fire_Act[A].Tipe<>0) and (Fire_Act[A].Tipe<6) then
```

Jika tipe dari FIRE_ACT <> 0 dan lebih kecil dari 6 maka lakukan proses di bawah ini. FIRE_ACT akan memiliki tipe = 0 jika peluru tersebut telah menabrak sesuatu dan non-aktif.

```
          Begin
            X:=X+Vx;
            Y:=Y+Vy;
```

Tambahkan semua posisi peluru dengan kecepatannya.

```
            If (X>Xp+320) or (X<Xp) or (Y>Yp+200) or (Y<Yp) then
              Begin
                Fire_Act[A].Tipe:=0;
```

Jika peluru tersebut melebihi batas layar maka non-aktifkan.

```
        End;
      If Collision(1,X,Y,PicSize[Pic].Length,;
          PicSize[Pic].Width) then
        Begin
          Make_Anim(Tipe,X,Y);
          Fire_Act[A].Tipe:=0;
```

Jika terjadi tabrakan antara peluru dengan balok yang aktif maka non- aktifkan peluru tersebut dan aktifkan animasi dengan tipe yang sama dengan tipe peluru. Dengan cara ini animasi yang dihasilkan akan sesuai dengan gambar peluru yang dikirimkan.

```
          End;
      End;
With My_Player do
  If Active then
    Begin
```

Jika si Buggy masih hidup maka ...

```
    If Collision(1,X,Y+1,PicSize[1].Length,;
        PicSize[1].Width)=False then
      Begin
        If Vy<9 then Vy:=Vy+1;
```

Deteksi apakah di bawah kaki si Buggy ada tempat berpijak atau tidak. Jika tidak, lakukan perhitungan gravitasi terhadap kecepatan vertikal si Buggy. Batasi terminal velocity si Buggy pada 9 pixel per loop.

```
      End;

Tipe:=Check_Move(1,X,Y,Vx,Vy,Rx,Ry,PicSize[1].Length,PicSize[1].Widt
h);
```

Deteksi bagian dari si Buggy yang menabrak balok.

```
    Case Tipe of
      0:Begin
          { No Collision }
          If Collision(1,X+Rx,Y+Ry,PicSize[1].Length,;
                PicSize[1].Width) = False then
            Begin
              X:=X+Rx;Y:=Y+Ry;
```

Jika tidak terjadi tabrakan maka pindahkan koordinat si Buggy.

```
            End;
        End;
      1:Begin
          { Side Collision }
          Vx:=0;
```

Jika terjadi tabrakan samping maka set Vx = 0.

```
If Collision(1,X+Vx,Y+Vy,PicSize[1].Length,;
      PicSize[1].Width) = False then
```

Pastikan bahwa dengan koordinat yang baru tidak terjadi tabrakan dengan balok.

```
Begin
  X:=X+Vx;Y:=Y+Vy;
```

Pindahkan posisi si Buggy. Karena Vx = 0 maka si Buggy tidak akan berpindah secara horizontal.

```
    End;
  End;
4:Begin
    { Upper Collision }
    If Collision(1,X+Rx,Y+Ry,PicSize[1].Length,;
          PicSize[1].Width) = False then
```

Pastikan bahwa dengan koordinat yang baru tidak terjadi tabrakan dengan balok.

```
Begin
  Y:=Y+Ry;
  X:=X+Rx;
```

Terjadi tabrakan atas maka pindahkan posisi si Buggy.

```
  End;
Vy:=-Vy;
```

Set Vy = -Vy sehingga si Buggy terpantul ke bawah.

```
  End;
2:Begin
    { Bottom Collision }
    Status:=1;
    FCtr:=1;
    Vy:=0;
```

Terjadi tabrakan bawah berarti si Buggy telah menginjak permukaan balok/tanah. Maka set status = 1 yang berarti si Buggy berada pada situasi berdiri. Set Frame counter = 1 dan kecepatan sumbu Y = 0.

```
If Collision(1,X+Rx,Y+Ry,PicSize[1].Length,;
      PicSize[1].Width) = False then
```

Pastikan dengan koordinat yang baru, si Buggy tidak menabrak balok.

```
            Begin
              X:=X+Rx;Y:=Y+Ry;
```

Pindahkan posisi si Buggy.

```
            End;
        End;
    3:Begin;
```

Terjadi tabrakan dua sisi.

```
        If Vy>=0 then
          Begin
            Status:=1;
            Fctr:=1;
```

Jika kecepatan sumbu Y positif maka anggap si Buggy telah berpijak pada permukaan balok/tanah.

```
            End;
        If Collision(1,X+Rx,Y+Ry,PicSize[1].Length,;
              PicSize[1].Width) = False then
```

Pastikan bahwa si Buggy berada pada koordinat yang tidak menabrak balok.

```
            Begin
              X:=X+Rx;Y:=Y+Ry;
```

Pindahkan posisi si Buggy.

```
            End;
        Vy:=0;Vx:=0;
```

Set Vx dan Vy = 0 karena terjadi tabrakan dua sisi.

```
          End;
      End;
    End;
For A:=1 to 40 do
With Object_Act[A] do
```

Lakukan proses yang sama terhadap musuh.

```
If Ctr5=0 then
Begin
```

Jika Ctr5=0 yang berarti musuh masih hidup.

```
  Xl:=Object_Char[Enemy_Spot[Obj_Tagged].Tipe].Length;
  Yl:=Object_Char[Enemy_Spot[Obj_Tagged].Tipe].Width;
```

Ambil panjang dan lebar musuh.

```
If Obj_Tagged<>0 then
  Begin
    If Collision(1,X,Y+1,XL,YL)=False then
```

Jika tidak ada balok yang berfungsi sebagai tempat berpijak objek maka ...

```
Begin
  Case Enemy_Spot[Obj_Tagged].Tipe of
    1:;
    2:;
    3:;
    4:Begin
        Vx:=-Vx;
        If Direction=1 then Direction:=0 else ;
            Direction:=1;
        If Collision(1,X+Vx,Y+1,XL,YL)=False then
          If Vy<9 then Vy:=Vy+1;
```

Jika objek tersebut ialah Shower Man maka ubah arah kecepatan horizontal. Jika setelah pada frame berikutnya tetap masih tidak ada tempat berpijak, lakukan sistem gravitasi sehingga objek ini jatuh ke bawah.

```
      End;
    5:Begin
        If Vy<9 then Vy:=Vy+1;
      End;
    6:Begin
        If Vy<9 then Vy:=Vy+1;
      End;
    7:;
    8:Begin
        If Vy<9 then Vy:=Vy+1;
      End;
```

Untuk tipe 5, 6, 8 dilakukan sistem gravitasi.

```
    End;
  End;
Tipe:=Check_Move(1,X,Y,Vx,Vy,Rx,Ry,XL,YL);
```

Deteksi bagian yang tertabrak objek.

```
Case Tipe of
  0:Begin
      { No Collision }
      X:=X+Rx;Y:=Y+Ry;
    End;
  1:Begin
      { Side Collision }
      Vx:=-Vx;
      If Direction=1 then Direction:=0 else Direction:=1;
      X:=X+Vx;Y:=Y+Vy;
```

```
        End;
      4:Begin
          { Upper Collision }
          Y:=Y+Ry;
          X:=X+Rx;
          Vy:=-Vy;
        End;
```

Untuk tipe tabrakan nomor 0, 1, 4 sama seperti pada pendeteksian tabrakan untuk si Buggy.

```
      2:Begin
          { Bottom Collision }
          Vy:=0;
          X:=X+Rx;Y:=Y+Ry;
          Case Enemy_Spot[Obj_Tagged].Tipe of
            5:Begin
                Vy:=-8;
                Ctr1:=1;
```

Jika terjadi tabrakan bawah terhadap Peer Doll maka pantulkan dia ke atas yaitu dengan mengisi kecepatan vertikal awalnya dengan -8.

```
              End;
            8:Begin
                Ctr2:=1;
                Ctr3:=0;
                Vx:=0;
```

Jika Teeth yang menabrak maka isikan Vx = 0.

```
              End;
          End;
        End;
      3:Begin;
          Vy:=0;Vx:=0;
        End;
    End;
  End;
End Else
Begin
  Inc(Y,3);
  If Y>Yp+200 then
    Begin
      Enemy_Spot[Obj_Tagged].Tipe:=0;
      Obj_Tagged:=0;
    End;
```

Jika objek sedang dalam tahap mati (jatuh ke bawah layar) maka tambahkan terus koordinat Y dari objek dengan 3 hingga akhirnya menembus batas bawah layar. Jika hal ini terjadi, lenyapkan objek tersebut dari data objek dan dari buffer OBJEK_ACT.

```
      End;
  End;

Procedure Object_Collision;
Var A,B:Integer;
    XL,YL,XM,YM:Integer;
  Begin
   For A:=21 to 100 do
     If (Fire_Act[A].Tipe<>0) And (My_Player.Ctr5>30) and;
        (My_Player.InVul=False) then
       Begin
```

Deteksi tabrakan antara peluru musuh dengan si Buggy.
Tabrakan diperbolehkan jika Buggy dalam kondisi tidak kebal.

```
         XL:=Object_Char[1].Length;
         YL:=Object_Char[1].Width;
         XM:=PicSize[Fire_Act[A].Pic].Length;
         YM:=PicSize[Fire_Act[A].Pic].Width;
```

Ambil panjang dan lebar Buggy dan peluru yang akan dicek.

```
       If (Fire_Act[A].X+XM>My_Player.X) and (Fire_Act[A].;
             X<My_Player.X+XL) and
          (Fire_Act[A].Y+YM>My_Player.Y) and (Fire_Act[A].;
             Y<My_Player.Y+YL) then
           Begin
```

Jika peluru tersebut menabrak si Buggy maka ...

```
                With My_Player do
                  Begin
                    Status:=4;
                    FCtr:=1;
                    Ctr1:=1;
                    My_Player.Vy:=-6;
                    Ctr5:=1;
                    Create_Message('1 LIVE LOST',My_Player.X-20,;
                           My_Player.Y-20,LightRed);
                    Dec(Life);
```

Aktifkan message dan kurangi nyawa si Buggy. Set status si Buggy = 4.

```
                    If Life=0 then
                      Begin
                        CountDown:=True;
```

Jika nyawa telah habis maka aktifkan CountDown untuk melakukan hitungan 30 loop sebelum program berhenti.

```
                        End;
                    End;
              Fire_Act[A].Tipe:=0;
```

Matikan peluru.

```
Make_Anim(1,Fire_Act[A].X,Fire_Act[A].Y);
```

Ciptakan animasi percikan peluru.

```
            A:=100;
        End;
    End;
For A:=1 to 20 do
  If Fire_Act[A].Tipe<>0 then
```

Deteksi tabrakan antara tembakan si Buggy dengan musuh.

```
      For B:=1 to 40 do
        With Object_Act[B] do
          If (Obj_Tagged<>0) and (Ctr5=0) and (Enemy_Spot;
              [Obj_Tagged].Tipe<9) then
            Begin
```

Jika musuh memiliki tipe < 9 yang berarti objek tersebut musuh bukannya bonus dan masih hidup maka ...

```
XL:=Object_Char[Enemy_Spot[Obj_Tagged].Tipe].Length;
YL:=Object_Char[Enemy_Spot[Obj_Tagged].Tipe].Width;
XM:=PicSize[Fire_Act[A].Pic].Length;
YM:=PicSize[Fire_Act[A].Pic].Width;
```

Hitung panjang dan lebar objek dan peluru si Buggy.

```
If (Fire_Act[A].X+XM>X) and ;
      (Fire_Act[A].X<X+XL) and
   (Fire_Act[A].Y+YM>Y) and ;
      (Fire_Act[A].Y<Y+YL) then
     Begin
```

Jika terjadi tabrakan maka ...

```
If (Enemy_Spot[Obj_Tagged].Tipe=3) and;
       (Ctr2<30) then else
Begin
```

Jika musuh yang ditembak bukan Piranha Plant (Tipe = 3 ialah Piranha Plant) yang pada saat bersembunyi di dalam tanah (Ctr2<30 menunjukkan saat Piranha Plant bersembunyi) maka ...

```
Ctr5:=1;
Ctr1:=1;
Make_Anim(4,Fire_Act[A].X,Fire_Act[A].Y);
```

Aktifkan animasi ledakan merah.

```
                    Fire_Act[A].Tipe:=0;
                    Create_Message('100 PTS',Fire_Act[A].X-20,;
                          Fire_Act[A].Y-20,15);
                    Inc(My_Player.Score,100);
```

Matikan peluru.
Tampilkan message.
Tambah score si Buggy.

```
                     B:=40;
                    End;
                   End;
              End;
  If (My_Player.Ctr5>30) then
```

Jika si Buggy tidak dalam masa kebal maka ...

```
  For A:=1 to 40 do
    With Object_Act[A] do
     If (Obj_Tagged<>0) and (Ctr5=0) then
      Begin
```

Jika objek masih hidup maka ...

```
        XL:=Object_Char[Enemy_Spot[Obj_Tagged].Tipe].Length;
        YL:=Object_Char[Enemy_Spot[Obj_Tagged].Tipe].Width;
        XM:=Object_Char[1].Length;
        YM:=Object_Char[1].Width;
```

Ambil panjang dan lebar Buggy dan objek.

```
        If (My_Player.X+XM>X) and (My_Player.X<X+XL) and
           (My_Player.Y+YM>Y) and (My_Player.Y<Y+YL) then
             Begin
```

Jika terjadi tabrakan maka ...

```
            If (Enemy_Spot[Obj_Tagged].Tipe=3) and ;
                   (Ctr2<30) then else
            Begin
```

Jika objek tersebut bukan Piranha Plant yang dalam masa bersembunyi maka ...

```
            Case Enemy_Spot[Obj_Tagged].Tipe of
              1..8:Begin
              If (My_Player.Vy>0) and (Enemy_Spot[Obj_Tagged].;
                    Tipe<>3) and
              (Enemy_Spot[Obj_Tagged].Tipe<>6) or ;
                   My_Player.Invul then
```

Jika objek yang diperiksa bukan bonus dan si Buggy sedang dalam masa jatuh dan objek yang ditimpa dari atas bukan Piranha Plant dan juga bukan Red Wheel atau si Buggy sedang dalam masa kebal maka ...

```
Begin
  Ctr5:=1;
  Ctr1:=1;
  My_Player.FCtr:=1;
  My_Player.Vy:=-6;
  Create_Message('100 PTS',My_Player.X-20,;
    My_Player.Y-20,15);
  Inc(My_Player.Score,100);
```

Tambah score, tampilkan message, lompatkan si Buggy dengan mengatur Vy = -6.

```
End Else
With My_Player do
  Begin
    Status:=4;
    FCtr:=1;
    Ctr1:=1;
    My_Player.Vy:=-6;
    Ctr5:=1;
    My_Player.PowerUp:=False;
    Create_Message('1 LIVE LOST',My_Player.X-20,;
          My_Player.Y-20,LightRed);
    Dec(Life);
    If Life=0 then
      Begin
        Finish:=True;
      End;
```

Selain itu si Buggy tertabrak musuh. Kurangi nyawa, tampilkan message dan lain-lainnya seperti di atas.

```
      End;
End;
10:Begin
    My_Player.PowerUp:=True;
    Create_Message('GOT MISSILE LAUNCHER',;
       My_Player.X-20,My_Player.Y-20,White);
    Inc(My_Player.Misile,10);
    Enemy_Spot[Object_Act[A].Obj_Tagged].Tipe:=0;
    Object_Act[A].Obj_Tagged:=0;
```

Jika objek yang ditabrak ialah misile launcher maka set POWERUP = TRUE. Tambahkan total Misile dengan 10. Non aktifkan objek yang ditabrak.

```
  End;
11:Begin
    Inc(My_Player.Misile,10);
```

```
                Create_Message('GOT 10 ROCKETS',My_Player.X-20,;
                    My_Player.Y-20,Yellow);
                Enemy_Spot[Object_Act[A].Obj_Tagged].Tipe:=0;
                Object_Act[A].Obj_Tagged:=0;
```

Jika yang ditabrak ialah rocket maka tambah rocket dengan 10, tampilkan message dan non-aktifkan objek yang ditabrak.

```
              End;
            12:Begin
                My_Player.InVul:=True;
                My_Player.InCtr:=400;
                Make_Anim(6,My_Player.X-10,My_Player.Y-10);
                Create_Message('INVULNARBILITY ON',;
                    My_Player.X-20,My_Player.Y-20,Yellow);
                Enemy_Spot[Object_Act[A].Obj_Tagged].Tipe:=0;
                Object_Act[A].Obj_Tagged:=0;
```

Jika yang ditabrak ialah kekebalan maka set INVUL = TRUE dan INCTR = 400 yang berarti Buggy akan kebal selama 400 loop. Tampilkan animasi bola kekebalan. Tampilkan message dan hilangkan objek yang ditabrak.

```
              End;
            13:Begin
                Inc(My_Player.Score,100);
                Create_Message('100 PTS',My_Player.X-20,;
                    My_Player.Y-20,Yellow);
                Enemy_Spot[Object_Act[A].Obj_Tagged].Tipe:=0;
                Object_Act[A].Obj_Tagged:=0;
```

Jika yang ditabrak ialah koin maka tampilkan message, tambah score dan hilangkan objek yang ditabrak.

```
              End;
          End;
          A:=40;
        End;
      End;
    End;
End;

Procedure Do_Status;
Var A,B:Integer;
    R,Dx,Dy:Real;
  Begin
    For A:=1 to 100 do
      With Anim_Act[A] do
        If Active then
          Begin
            Inc(Ctr);
```

Tambahkan frame counter dari animasi.

```
Case Tipe of
  1:If Ctr>12 then Active:=False;
  2:If Ctr>6 then Active:=False;
  3:If Ctr>12 then Active:=False;
  4:If Ctr>12 then Active:=False;
  5:If Ctr>15 then Active:=False;
  6:Begin
      If Ctr>8 then Ctr:=1;
      If My_Player.Invul=False then Active:=False;
    End;
  Else Active:=False;
```

Karena tiap tipe animasi memiliki jumlah frame yang berbeda-beda maka matikan animasi jika frame counter telah mencapai maksimum frame dari tiap jenis.

Animasi tipe 6 agak lain karena animasi tersebut aktif jika Buggy masih memiliki bola kekebalan. Untuk itu dilakukan pendeteksian terhadap variabel INVUL.

```
      End;
    End;
For A:=1 to 100 do
  With Fire_Act[A] do
    If Tipe<>0 then
      Begin
        If A<=20 then
          Begin
            If Round(Random(2))=1 then;
                  Make_Anim(5,X+Round(Random(3))-2,
            Y+Round(Random(3))-2);
            If Round(Random(2))=1 then;
                  Make_Anim(5,X+Round(Random(3))-2,
            Y+Round(Random(3))-2);
          End;
```

Tampilkan animasi asap secara random di sekeliling ekor peluru bernomor lebih kecil dari 21 yang berarti peluru si Buggy.

```
        Inc(Ctr1);
```

Tambahkan frame counter.

```
      End;
For A:=1 to 20 do
  If Message[A].Active then
    With Message[A] do
      Begin
        Dec(Y,2);
        Inc(Ctr);
        If Ctr>15 then Active:=False;
```

Tambahkan Frame counter dari Message, naikkan koordinat Y sehingga message terlihat naik ke atas.
Jika message telah muncul selama 15 loop maka matikan message.

```
        End;
With My_Player do
Begin
  If InVul then
    Begin
      Dec(InCtr);
```

Kurangi selalu counter dari kekebalan.

```
      If InCtr<1 then
        Begin
          InVul:=False;
```

Jika counter kekebalan habis maka set INVUL = FALSE.

```
        End;
    End;
  Dec(Ctr3);
  If Ctr3<1 then Ctr3:=0;
```

Ctr3 adalah selang waktu di mana si Buggy masih belum bisa menembak. Untuk itu pada setiap loop, variabel ini dikurangi hingga 0. Jika Ctr3 berharga 0 maka si Buggy dapat menembak lagi. Hal ini dilakukan agar si buggy tidak dapat melakukan tembakan beruntun pada setiap loopnya.

```
  If Ctr5>0 then
    Begin
      Inc(Ctr5);
      If Ctr5>30 then Ctr5:=32;
    End;
If Not Flag and Not Flag2 then
  Begin
    If Vx<0 then Vx:=Vx+1;
    If Vx>0 then Vx:=Vx-1;
  End;
```

Jika tidak ditekannya tombol kiri atau kanan maka kurangi kecepatan horizontal secara momentum.

```
  Case Status of
    2:Begin
        Inc(Fctr);
        If Fctr>18 then Fctr:=1;
        If Vx=0 then Status:=1;
```

Jika si Buggy sedang berjalan dan Frame Counter telah mencapai 18 maka kembalikan frame counter ke 1.

```
  End;
3:Begin
    Inc(Fctr);
    If Not Flag and Not Flag2 then
      Begin
        If Vx<0 then Vx:=Vx+1;
        If Vx>0 then Vx:=Vx-1;
      End;
    Inc(Ctrl);
    If Ctrl>=20 then Ctrl:=20;
```

Jika si Buggy sedang melompat maka jika tidak ditekannya tombol kiri atau kanan, kurangi kecepatan horizontal secara momentum.

Ctrl<12 digunakan untuk batas waktu di mana kecepatan awal dari lompatan si Buggy dapat ditahan. Untuk ini pada setiap loop Ctrl selalu ditambah hingga pada harga akhir = 20.

```
  End;
5:Begin
    Inc(FCtr);
    If FCtr>4 then
      Begin
        FCtr:=1;
        Status:=1;
      End;
```

Jika frame counter untuk animasi menembak telah habis maka kembalikan status si Buggy ke status = 1 yaitu berdiri.

```
    If Not Flag and Not Flag2 then
      Begin
        If Vx<0 then Vx:=Vx+1;
        If Vx>0 then Vx:=Vx-1;
      End;
  End;
4:Begin
    Inc(Ctrl);
    If Ctrl>3 then
      Begin
        Inc(Fctr);
        If Fctr>20 then
          Begin
            FCtr:=1;
            Status:=1;
          End;
        Ctrl:=0;
```

Jika frame counter si Buggy yang terlempar sudah habis maka kembalikan status si Buggy = 1 yaitu berdiri.

```
          End;
       End;
    End;
End;
For A:=1 to 40 do
  With Object_Act[A] do
    If Obj_Tagged<>0 then
     With Enemy_Spot[Obj_Tagged] do
      If Ctr5=0 then
      Begin
        Case Enemy_Spot[Obj_Tagged].Tipe of
          2:;
          3:Begin
              Inc(Ctr2);
```

Ini adalah status Piranha Plant. Ctr2 digunakan untuk saat di mana Piranha Plant sembunyi atau muncul dari permukaan tanah.

```
                If Ctr2>30 then
```

Jika piranha plant telah muncul maka ...

```
                  Begin
                    Inc(Ctr3);
```

Gunakan Ctr3 untuk memperlambat frame counter. Karena frame counter akan diupdate setiap Ctr3 > 6.

```
                  If Ctr3>5 then
                    Begin
                      Inc(Ctr1);
```

Update frame counter.

```
                      Ctr3:=0;
                      If Ctr1>15 then
                        Begin
                          Ctr2:=0;
                          Ctr1:=0;
                        End;
```

Jika frame counter telah mencapai 15 maka lakukan tahap sembunyi lagi.

```
                          End;
                      End;
                  End;
              4:Begin
                  Inc(Ctr2);
```

Ini adalah status Shower Man.
Ctr2 digunakan untuk memperlambat Frame Counter.

```
If Ctr2>=2 then
  Begin
    Ctr2:=0;
    Inc(Ctr1);
    If Ctr1>2 then Ctr1:=1;
```

Tambahkan Ctr1 sebagai frame counter.

```
  End;
Dec(Ctr3);
```

Ctr3 berfungsi sebagai delay dari proses menembak. Jika Ctr3 >=0 maka Shower Man belum boleh menembak.

```
If Ctr3<0 then
  Begin
    Ctr3:=Round(Random(20))+20;
```

Isi Ctr3 dengan delay yang baru yang didapat dari bilangan RANDOM antara 20-40.

```
For B:=21 to 100 do
  With Fire_Act[B] do
    If Tipe=0 then
      Begin
        fire_Act[B].Tipe:=2;
        X:=Object_Act[A].X+20;
        Y:=Object_Act[A].Y+3;
        Ctr1:=1;
        Dx:=My_Player.X-X;
        Dy:=My_Player.Y-Y;
        R:=Sqrt(Dx*Dx+Dy*Dy);
        If R=0 then R:=0.000001;
        Vx:=Round(Dx*4/R);
        Vy:=Round(Dy*4/R);
        Pic:=78;
```

Lakukan proses menembak ke arah Buggy. Proses perhitungan arah dilakukan untuk mendapatkan Vx dan Vy yang mengarah ke si Buggy. Teknik ini sama persis dengan tekink yang kita pakai pada pembuatan game FIRE FIGHTER dalam memproses tembakan musuh ke arah pesawat kita.

```
        B:=100;
      End;
  End;
End;
5:Begin
  Inc(Ctr1);
  If Ctr1>100 then Ctr1:=1;
End;
6:Begin
  Inc(Ctr1);
```

```
      If Ctr1>2 then Ctr1:=1;
    End;
7:Begin
    Inc(Ctr2);
```

Ini adalah status Cloud.

```
If Ctr2>3 then
  Begin
    If My_Player.X<Object_Act[A].X then
      Begin
        Direction:=0;
        Vx:=-3;
      End Else
      Begin
        Direction:=1;
        Vx:=3;
      End;
```

Lakukan pendeteksian letak si Buggy dan arahkan kecepatan si Cloud ke si Buggy.

```
Ctr2:=0;
Inc(Ctr1);
If Ctr1>2 then Ctr1:=1;
```

Update frame counter pada Ctr1.

```
Dec(Ctr3);
If (Ctr3<0) then Ctr3:=0;
```

Kurangi delay penembakan.

```
If (Ctr3=0) then
  Begin
    Ctr3:=Round(Random(20))+20;
```

Hasilkan delay penembakan yang baru antara 20-40 loop.

```
For B:=21 to 100 do
  With Fire_Act[B] do
    If Tipe=0 then
      Begin
        Fire_Act[B].Tipe:=3;
        X:=Object_Act[A].X+10;
        Y:=Object_Act[A].Y+25;
        Fire_Act[B].Ctrl:=1;
        Vx:=0;
        Vy:=4;
        Pic:=74;
        B:=100;
```

Lakukan proses menembak. Adapun arah tembakan ini ialah ke bawah sebesar 4 pixel per loop.

```
                    End;
                End;
            End;
        End;
    8:Begin
        Inc(Ctr4);
```

Ini adalah status dari Teeth.
Ctr4 digunakan untuk memperlambat frame counter.

```
If Ctr4>3 then
  Begin
    Ctr4:=0;
    Inc(Ctr1);
    If Ctr1>4 then Ctr1:=1;
```

Update frame counter pada Ctr1.

```
Inc(Ctr2);
```

Tambahkan Ctr2 yang menunjukkan saat di mana Teeth akan melompat.

```
If (Ctr2>30) and (Ctr3=0) then
  Begin
```

Jika Ctr2>30 maka Teeth akan diizinkan melompat. Ctr3 = 0 berarti si Teeth masih belum melakukan proses lompat. Jika proses lompat dilakukan maka Ctr3 = 1. Tujuan dari pendeteksian Ctr3 = 0 ini ialah supaya si Teeth tidak melompat pada saat dia sedang dalam situasi melompat.

```
Ctr1:=3;
Ctr3:=1;
```

```
Set Ctr3 =1.
```

```
Object_Act[A].Vy:=-8;
```

Isikan kecepatan vertikal dengan -8 sehingga dia akan melompat ke atas.

```
If My_Player.X<Object_Act[A].X then
  Begin
    Vx:=-4;
    Direction:=0;
  End Else
  Begin
    Vx:=4;
    Direction:=1;
  End;
```

Arahkan kecepatan horizontal ke arah si Buggy.

```
                    End;
                  End;
              End;
          10,11,12,13:Begin
              Inc(Ctr4);
              If Ctr4>3 then
                Begin
                  Ctr4:=0;
                  Inc(Ctr1);
                  If Ctr1>3 then Ctr1:=1;
```

Status ini ialah status bonus.

```
              End;
            End;
          End;
        End;
  End;

Procedure OutTro;
Var A:Integer;
  Begin
    Set_All_Palette(User_Pal,63,0,0);
    For A:=1 to 50 do
      Begin
        Put_Parallax_Scroll(Xp,Yp,Process^,UnComp^);
        Redraw_Screen;
        Rll_UnCompress(Picture[18+((A Mod 8) div 4)]^,UnComp^);
        T_Put3D(Process^,160-A,100-A,A*2,A*2,UnComp^,PalTable^);
        MoveTo(Process^,Screen^);
        If A Mod 10 = 0 then Metamorph_Palette(Active_Pal,;
           User_Pal,1);
        Refresh_Palette;
      End;
    Set_All_Palette(User_Pal,0,0,0);
    For A:=1 to 63 do
      Begin
        Metamorph_Palette(Active_Pal,User_Pal,1);
        Refresh_Palette;
      End;
    FillChar(Screen^,64000,0);
    Active_Pal:=Default_Pal;
    Refresh_Palette;
    Put_Center(90,8,LightBlue,'GAME OVER !!!');
    Delay(2000);
```

Procedure ini dipanggil jika terjadi GAME OVER. Tujuannya ialah untuk menampilkan animasi penutup.

```
  End;

Procedure Main;
  Begin
    Repeat
      Detect_Keyboard;
```

Deteksi tombol keyboard.

```
      Scanning_Area;
```

Ambil balok dan objek yang dekat dengan Buggy dan masukkan ke buffer.

```
      If My_Player.X-Xp<110 then Xp:=My_Player.X-110;
      If My_Player.Y-Yp<50  then Yp:=My_Player.Y-50;
      If My_Player.X-Xp>190 then Xp:=My_Player.X-190;
      If My_Player.Y-Yp>162 then Yp:=My_Player.Y-162;
```

Lakukan penggeseran parallax scroll yang selalu mengikuti gerakan si Buggy.

```
      If Xp<0 then Xp:=0;
      If Yp<0 then Yp:=0;
      If Xp>2680 then Xp:=2680;
      If Yp>300 then Yp:=300;
```

Batasi parallax scroll.

```
      Put_Parallax_Scroll(Xp,Yp,Process^,UnComp^);
```

Letakkan parallax scroll pada virtual screen.

```
      Redraw_Screen;
      MoveTo(Process^,Screen^);
      Do_Status;
      Object_Collision;
```

Deteksi tabrakan antarobjek.

```
      Do_Physics;
```

Ubah posisi setiap objek.

```
      If CountDown then
        Begin
          Dec(CtrCD);
          If CtrCD=0 then Finish:=True;
        End;
```

Lakukan proeses delay akhir jika terjadi GAME OVER.

```
    Until Finish or (key=1);
```

Keluar jika tombol ESC ditekan atau game telah usai.

```
    OutTro;
  End;

Procedure Removal;
  Begin
    CloseVga256;
    SetIntVec($9,OldInt9);
  End;

Begin
  Init_Device;
  Init_Environment;
  Init_Variable;
  Init_Picture;
  Init_Mission;
  Main;
  Removal;
End.
```

YESSS!! Akhirnya selesailah pembahasan kita tentang game BUG HUNTER ini.

INFORMASI DAFTAR FILE

GCAP10.PAS	Program Resident yang berfungsi sebagai Screen Capture.
B_PUT1.PAS	Program untuk mencoba PutImage yang baru.
B_PUT2.PAS	Program untuk mencoba PutImage 1 warna.
B_TR1.PAS	Program untuk menghasilkan tabel Transparant.
B_TR2.PAS	Program untuk mencoba Put Transparant.
B_TR3.PAS	Program untuk mencoba Put Transparant 1 warna.
B_COMP1.PAS	Program untuk mencoba teknik Color Compression.
B_COMP2.PAS	Program untuk mencoba teknik Zero Compression.
B_COMP3.PAS	Program untuk mencoba teknik RLL Compression.
B_3D1.PAS	Program untuk mencoba Put Resize.
B_3D2.PAS	Program untuk mencoba Put Resize Transparant.
PCXTO64K.PAS	Program untuk menciptakan background scroll dari PCX file untuk proses horizontal scrolling.
PCXTOSRC.PAS	Program untuk menciptakan background scroll dari PCX file untuk proses parallax scrolling.
HORISON.PAS	Program untuk mencoba teknik Horizontal Scrolling.
PARALLAX.PAS	Program untuk mencoba teknik Parallax Scrolling.
B_COL1.PAS	Program untuk mencoba teknik tabrakan dengan pendeteksian pixel dengan kecepatan rendah.
B_COL11.PAS	Program untuk mencoba teknik tabrakan dengan pendeteksian pixel dengan kecepatan tinggi.

B_COL2.PAS	Program untuk mencoba teknik tabrakan dengan perhitungan vektor pada sumbu X.
B_COL3.PAS	Program untuk mencoba teknik tabrakan dengan perhitungan vektor pada sumbu X dan Y.
B_MOM1.PAS	Program untuk mencoba teknik scroll dengan kecepatan linear.
B_MOM2.PAS	Program untuk mencoba teknik scroll dengan kecepatan momentum.
B_MOM3.PAS	Program untuk menggerakkan sprite dengan kecepatan momentum.
B_KEY1.PAS	Program untuk mencoba teknik pendeteksian keyboard dengan interrupt $9 dari DOS.
B_KEY2.PAS	Program untuk mencoba teknik pendeteksian keyboard dengan interrupt $9 yang dirakit sendiri.
B_ENG15.PAS	Program engine dari BUG HUNTER.
B_EDITOR.PAS	Program editor dari BUG HUNTER.
VGACAD.PAS	Unit Animasi dan Graphic yang telah dimodifikasi.

PENUTUP

Dengan ini pembahasan animasi dan game profesional seri kedua telah selesai. Semoga apa yang telah diajarkan pada buku ini dapat membantu pembaca untuk mengembangkan teknik-teknik game programming.

Kita akan bertemu lagi pada kesempatan yang akan datang di mana akan dibahas teknik-teknik yang lebih canggih dan juga lebih menarik dari yang telah dibahas pada kesempatan ini. Untuk itu teknik yang telah diajarkan selama ini harus dikuasai terlebih dahulu karena tanpa dasar-dasar tersebut amatlah sulit untuk mempelajari teknik selanjutnya.

Terima kasih atas perhatian Anda sekalian dan sampai jumpa pada kesempatan yang akan datang.

Agustinus Nalwan

Clayton, Monash University

Victoria, Australia.

E-mail : ANAL1@MFS06.monash.edu.au

SERI APLIKASI PEMROGRAMAN

Pemrograman Animasi dan Game Profesional 2

Buku ini merupakan seri kedua SAP Pemrograman Animasi dan Game Profesional. Compiler yang dipergunakan di sini tetap sama yaitu kerja sama antara Turbo Pascal 7.00 dan Assembly.

Seri kedua ini akan membahas mengenai pembenahan pada unit graphic VGACAD,TPU. Dengan VGACAD,TPU yang baru ini, Anda kan mampu membuat game seperti Super Mario Bros, Prince of Persia, Street Fighter dan lain-lain. Selain itu Anda juga dapat menggunakannya untuk membuat game 3 Dimensi sederhana.

Pada akhir bab akan diberikan langkah-langkah pembuatan game BUG HUNTER yang bertipe SONIC, SUPER MARIO BROS & JAC JAZZ RABBIT dengan teknologi yang handal.

Penerbit PT Elex Media Komputindo
Jl. Palmerah Selatan 22
Jakarta 10270

Kelompok
Pemrograman
Ketrampilan
☑ Tingkat Pemula
☑ Tingkat Menengah
☐ Tingkat Mahir
Jenis Buku
☐ Referensi
☑ Tutorial
☐ Latihan

ISBN 979-637-830-2
12196157